Unity Pro

PL7 Application Converter
User Manual

07/2011

35006148.10

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

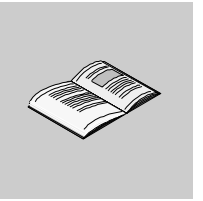
When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2011 Schneider Electric. All rights reserved.

Table of Contents



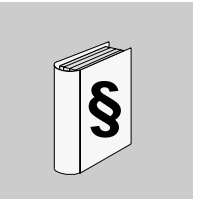
	Safety Information	**7**
	About the Book	**9**
Part I	**The PL7 application converter: general**	**11**
Chapter 1	**Overview of the converter**	**13**
	Overview of the PL7 Applications Converter	14
	Conversion Principle: General Points	16
	Conversion Principle: Applications and Processors	18
	Conversion principle: technical aspects	20
	Conversion to Unity ≥ 2.0	22
Part II	**PL7 application conversion procedure**	**25**
Chapter 2	**Conversion of a PL7 application**	**27**
	General	28
	Procedure for Converting a PL7 Application into a Unity Pro Application	30
	Results of the PL7 application conversion	32
	Conversion with the Conversion Wizard	33
	Topological Addresses of PL7 Micro in the Conversion Wizard	35
Chapter 3	**Conversion of a PL7 DFB**	**37**
	General	38
	Procedure for converting a PL7 DFB into Unity Pro	41
	Conversion of Protected DFBs	42
	Procedure for importing a PL7 DFB into Unity Pro	43
	Results of the PL7 DFB conversion	44
Chapter 4	**Analysis of a PL7 application converted into Unity Pro**	**45**
	General	46
	Analysis Procedure	47
	End of the analysis procedure	48
	'Converter error' message in the analysis procedure	49
	Miscellaneous errors in the analysis procedure	50
Part III	**Correspondence between PL7 and Unity Pro**	**51**
Chapter 5	**Inter-platform equivalence**	**53**
	Premium processors	54
	Atrium Processors	57

Chapter 6 Correspondence between application structures 59
6.1 Correspondences between PL7 and Unity Pro: structural elements 60
Tasks, events, and SRs 61
The sections 62
LD Sections Split into Small Sections 63
6.2 Correspondences between PL7 and Unity Pro: functional modules 64
Functional modules 64
Chapter 7 Correspondences between common language elements 65
7.1 Correspondences between PL7 and Unity Pro: types and tables 66
Types and arrays 67
Operations between mixed types. 69
7.2 Correspondences between PL7 and Unity Pro: language objects. 70
Immediate values. 71
Labels 74
System objects. 75
Memory objects (variables and constants) 77
Word bits 80
Symbolized Directly Addressed Variables 81
Symbolized tables and indexed objects. 82
SFBs 86
In-rack Input/Output objects. 89
Remote input/output objects 91
Grafcet objects. 93
7.3 Correspondences between PL7 and Unity Pro: instructions 95
Boolean instructions. 97
Comparison instructions 98
Bit table instructions. 100
Arithmetic instructions 101
Logic instructions 104
Shift instructions. 105
Numerical conversion instructions 107
Table instructions. 109
Character String Instructions 115
Time management instructions 116
Exchange Instructions 117
Input/output instructions. 118
Process control instructions 119
Other instructions. 120
Communication instructions. 121
TCP Open instructions. 122
Diagnostics instructions 124
Grafcet instructions 125
Human Machine Interface (HMI) instructions. 126

7.4 Correspondences between PL7 and Unity Pro: SFBs 127
Types of Unity Pro Function Block Instances of SFBs 128
Behavior of Converted SFBs in Modicon M340 Applications 129
Call of SFB Substitutions in Structured Text 130
Call of an SFB in instruction list language. 132
Call of an SFB in ladder language. 134
Chapter 8 Correspondences between ladder language elements. 137
The definition of a ladder network 138
The rungs 139
Coils 140
Operate blocks and compare blocks 141
Conversion restrictions: PL7 ladder language. 142
Chapter 9 Correspondences between Structured Text language elements 143
9.1 Correspondences between PL7 and Unity Pro: Structured Text language sequences 144
The sequences 144
9.2 Correspondences between PL7 and Unity Pro: Structured Text language instructions 145
Command instructions. 145
Chapter 10 Correspondences between Instruction List language elements 147
10.1 Correspondences between PL7 and Unity Pro: Instruction List language sequences 148
The sequences 148
10.2 Correspondences between PL7 and Unity Pro: Instruction List language instructions 149
Command instructions. 150
Boolean instructions 151
Instruction List language extensions 152
Chapter 11 Correspondences between Grafcet language elements 157
Grafcet instructions 158
Conversion Settings for PL7 Grafcet Elements. 159
Conversion restrictions: PL7 Grafcet language. 160
Chapter 12 Other correspondences between PL7 and Unity Pro elements 161
Printouts, animation tables, and runtime screens 161
Part IV Differences between PL7 and Unity Pro 163
Chapter 13 Differences between the application structures 165
13.1 Differences between PL7 and Unity Pro: functional modules 165
Functional Modules 165

Chapter 14 Differences between common language elements 167
14.1 Differences between PL7 and Unity Pro: types and tables 168
Types and tables 169
Operations between mixed types 170
14.2 Differences between PL7 and Unity Pro: objects 171
Immediate values 172
Memory objects (variables and constants) 173
Word bits 174
Symbolized tables and indexed objects 175
14.3 Differences between PL7 and Unity Pro: instructions and functions 176
Table instructions and functions 177
Process control, Other and Communication instructions 179
14.4 Differences between PL7 and Unity Pro: SFBs 181
Types of Unity Pro EFB instances 181
14.5 Differences between PL7 and Unity Pro: Timers 182
Differences between PL7 and Unity Pro: Timers 182
Chapter 15 Differences between Structured Text language elements 183
15.1 Differences between PL7 and Unity Pro: Structured Text language instructions 183
Command instructions 183
Chapter 16 Differences between Instruction List language elements 185
16.1 Differences between PL7 and Unity Pro: Instruction List language instructions 185
Boolean instructions 185
Chapter 17 Different display in runtime screens 187
Runtime screens 187
Appendices 189
Appendix A Recommendations 191
Recommendations during conversion 191
Glossary 197
Index 199

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

CAUTION

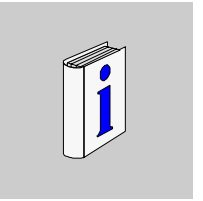
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This manual presents the PL7 application converter and describes the procedure for converting PL7 applications into Unity Pro applications. It also contains correspondence tables between PL7 programming elements and those of Unity Pro programming.

Validity Note

This documentation is valid for Unity Pro from version 6.0.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

The PL7 application converter: general

I

Overview of the converter

1

Subject of this Chapter

This chapter presents the PL7 application converter and describes the procedure for converting PL7 applications into Unity Pro applications.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Overview of the PL7 Applications Converter	14
Conversion Principle: General Points	16
Conversion Principle: Applications and Processors	18
Conversion principle: technical aspects	20
Conversion to Unity ≥ 2.0	22

Overview of the PL7 Applications Converter

General Points

The PL7 application conversion tool is integrated into Unity Pro, and enables PL7 applications to be converted into Unity Pro applications. In order to perform a conversion, you first need to:

- update the application to **PL7 V4.3** or higher,
- unprotect, where necessary, the application, as well as all sections, functional modules and DFBs,
- export and save the source file.

Application Conversion as a Whole

To convert an application as a whole, while keeping the same PLC family and selection of application parts or remapping of I/O objects is not needed, you have to use the PL7 Application Converter directly via the Unity Pro menu **File** →**Open**.

NOTE: Use the **Conversion Wizard** to convert PL7 Micro applications to Unity Pro M340 applications (see next paragraph).

Partial Application Conversion

To convert an application partially and/or the PLC family must be changed or remapping of I/O objects is needed, use the conversion wizard via the Unity Pro menu **Tools** →**Convert Partially**.

For detailed information, please refer to the *Operating Modes Manual -> Conversion Wizard*.

Conversion Principle

The PL7 application converter transforms the source files exported by PL7 ≥V4.3 into Unity Pro source files. Conversion stops automatically if the source file is from a earlier version of PL7 than 4.0 or if the configuration has not been exported *(see page 18)* (version earlier than V4.3).

Conversion is:

- called automatic when a complete application is converted,
- called semi-automatic when one or more DFBs are converted.

If the conversion is automatic, the software generates a source file that can be directly analyzed using Unity Pro. The application is imported automatically; after the Unity Pro project analysis is manually started, the output window *(see page 49)* containing the list of conversion errors is displayed on the screen.

If the conversion is semi-automatic, the converter generates a source file and a conversion report file.

In order for the contents of the converted source file to be usable, it needs to be imported **manually** into an application. Following this import; the output window *(see page 49)* containing the list of conversion errors is displayed on the screen.

⚠ WARNING

UNEXPECTED APPLICATION BEHAVIOR

The PL7 application converter translates the application but does not ensure its correct operation. Test the application after the conversion.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Conversion Principle: General Points

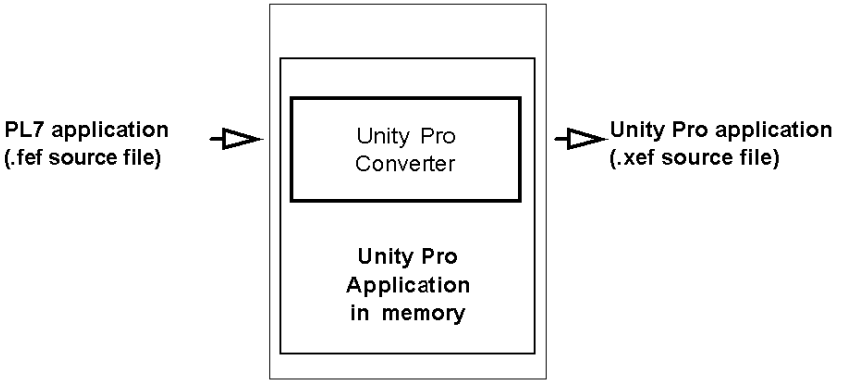
Introduction

The PL7 application converter is used to convert:

- a complete PL7 application. This procedure is called automatic,
- a PL7 DFB. This procedure is called semi-automatic.

Automatic Conversion

The procedure for converting a PL7 application into a Unity Pro is as follows:



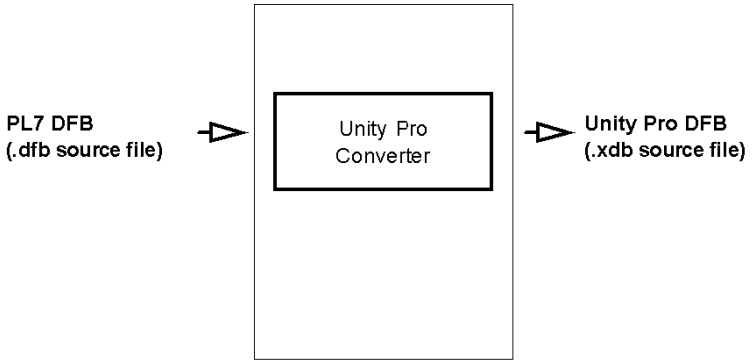
The **.fef** PL7 source file is converted into a **.xef** Unity Pro source file, then imported and analyzed automatically in the Unity Pro project. The analysis phase must be started manually in order to detect any conversion errors and display them on the screen in the form of an output window.

At the end of the procedure, the converted PL7 application and output window are opened and displayed on the screen in the Unity Pro software.

To correct any conversion errors, click on the error line displayed in the output window to go directly to the part of the program to be modified *(see page 49)*.

Semi-Automatic Conversion

The procedure for converting a PL7 DFB is as follows:



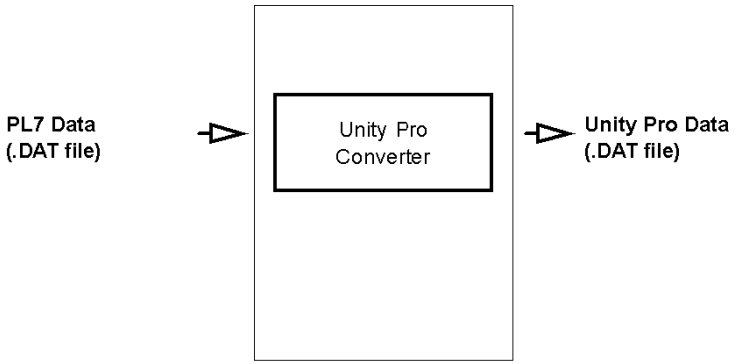
The **.dfb** PL7 source file is converted into a **.xdb** Unity Pro source file.

At the end of the procedure, the converted PL7 DFB is saved in its source format. In order for this DFB to be exploitable by Unity Pro, it must be **manually** imported into a Unity Pro application.

Following this import, you must start the project's analysis phase manually in order to detect any conversion errors and display them on the screen in the form of an output window.

To correct any conversion errors, click on the error line displayed in the output window to go directly to the part of the program to be modified *(see page 49)*.

The procedure for converting a PL7.DAT file is as follows :



The .dat PL7 file is converted into a .dat Unity Pro file by adding "_convert" in the file name. At the end of the procedure, the converted PL7 .dat file is saved. Then it is usable inside Unity Pro using the command "Transfert data from file to PLC" in the menu PLC.

Conversion Principle: Applications and Processors

General Points

The PL7 application converter transforms PL7 source files (.fef, .dfb) into Unity Pro source files (.xef, .xdb) and, during the conversion of a complete application (.fef), associates an equivalent to the old processor.

Software Applications

The conversion of a complete application is called automatic.

At the end of conversion, and after a manual analysis is started, the user has:

- a source file that is directly exploitable by Unity Pro,
- a Unity Pro application in memory,
- a conversion report file including all data, warnings and errors relating to the conversion,
- an output window containing the list of conversion errors.

NOTE: The recommended version of PL7 is ≥V4.3. However, you may, under your own responsibility, convert the exported applications with versions V4.0, V4.1 and V4.2 if the hardware configuration has been explicitly exported in the FEF file. To export the hardware configuration, you must modify the `PL7SYS.INI` file located in the WINNT or Windows folder on your PC. This file must contain the following two lines:

```
[PL7TOOL132]
```

`ExportConf=True` (to export configuration)

`ExportPl7Converter=True` (to export protected DFB)

The conversion of a DFB PL7 is called semi-automatic.

At the end of conversion, the user has:

- the source file of the converted DFB,
- a conversion report file including all data and warnings relating to the conversion.

Following a manual import of this converted DFB and the analysis in a Unity Pro project, the output window containing the list of conversion errors is displayed.

NOTE: The recommended version of PL7 is ≥V4.3 However, you may, under your own responsibility, convert the exported DFB with versions V4.0, V4.1 and V4.2

⚠ WARNING
UNEXPECTED APPLICATION BEHAVIOR The conversion of complete applications, or DFBs, must be done with a PL7V4.3 (or more recent) version only. Test the applications and DFBs after the conversion. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Processors

Sometimes the conversion procedure requires the size of the converted application to be increased.

By default, the PL7 application converter automatically updates the 'processor + memory card' configuration, and proposes an equivalent configuration *(see page 53)*. However, this default selection may be modified.

NOTE: All PL7 applications earlier than version V4.0 must be updated.

For applications managed by the following processors, the update procedure involves the compulsory replacement of the processor:

- Premium processor *(see page 54)* **TSX P57 •0**, **TSX P57 ••2** or **T PMX P57 ••2**,
- Atrium processor *(see page 57)* **T PCX 57 •••2**.

Conversion principle: technical aspects

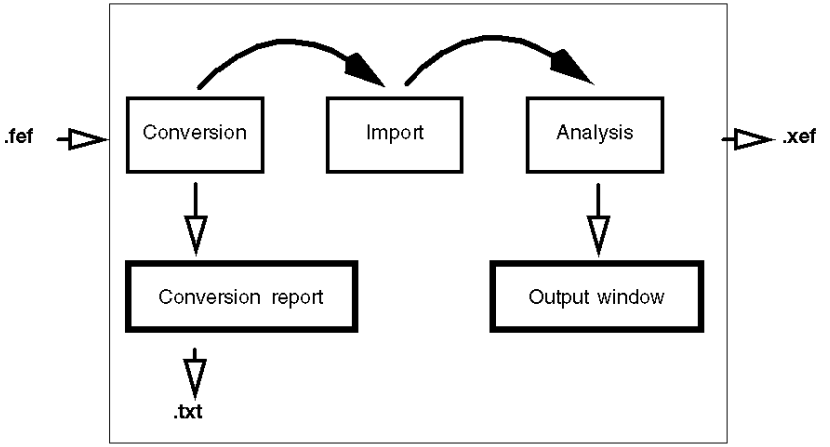
Introduction

The PL7 application converter is used to convert:

- a complete PL7 application; the procedure is automatic,
- a PL7 DFB; the procedure is semi-automatic.

Automatic conversion

The following diagram shows the technical aspects of the conversion procedure for a complete PL7 application.

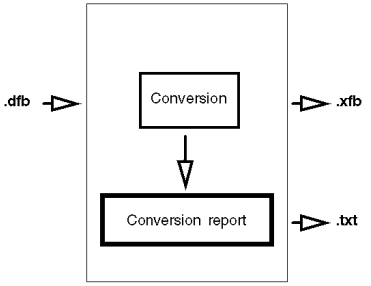


The following table describes the different steps involved in automatic conversion.

Step	Description
Conversion	During this phase, the **.fef** PL7 source file is converted into a **.xef** Unity Pro source file. Any data and warnings relating to the conversion are generated.
Conversion report	A **.txt** conversion report file is generated. It contains all data, warnings and errors relating to the conversion procedure.
Import	The **.xef** source file is imported automatically into Unity Pro.
Analysis	The imported file is analyzed by Unity Pro. Any conversion errors are detected and generated. **Note**: You must manually launch this analysis phase.
Output window	Any conversion errors are shown in the output window *(see page 48)* which is automatically displayed at the end of conversion.

Semi-automatic conversion

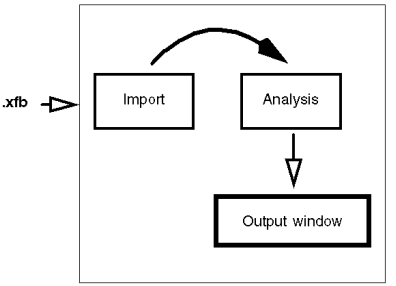
The following diagrams show the technical aspects of the conversion procedure for a PL7 DFB.



The following table describes the different steps involved in semi-automatic conversion.

Step	Description
Conversion	During this phase, the **.dfb** PL7 source file is converted into a **.xfb** Unity Pro source file. Any data and warnings relating to the conversion are generated.
Conversion report	A **.txt** conversion report file is generated. It contains all data and warnings relating to the conversion procedure. Once conversion is complete, the PL7 application converter stops.

The following diagram shows the technical aspects of the procedure for manually importing a converted PL7 DFB into a Unity Pro application.



The following table describes the different steps in the manual import procedure.

Step	Description
Import	In order for the converted **.xfb** source file to be exploitable by Unity Pro, it must be manually imported into a Unity Pro application.
Analysis	The imported file is analyzed by Unity Pro. Any conversion errors are detected. **Note**: You must manually launch this analysis phase.
Output window	Any conversion errors are shown in the output window *(see page 48)* which is automatically displayed at the end of analysis.

Conversion to Unity ≥ 2.0

At a Glance

The conversion of a PL7 application to a Unity Pro project may be achieved from version ≥4.3 of PL7.

However, you can convert ≥V4.4 PL7 applications to Unity Pro V1.0 or to Unity V2.0.

For the conversion of a ≥V4.4 PL7 to Unity Pro V1.0:

- If the functionalities or modules do not exist in V1.0, they will be indicated and not acknowledged in the new Unity project.
- If all functionalities or modules are available in Unity V1.0, the conversion is performed normally.

For a ≥V4.4 PL7 conversion to Unity Pro V2.0, the new functionalities and new modules are converted. The following paragraphs describe the new features and conversion rules to be used.

Fipio Applications

The PL7 applications containing the functionalities that implement a Fipio bus are converted form V4.3 or V4.4 or V4.5 PL7 to V2.0 Unity Pro.

The Fipio I/O objects are converted in line with the new topological addressing rule *(see page 91)*.

Lexium EF

PL7 applications containing Lexium EFs LXM_SAVE and LXM_RESTORE *(see page 117)* are converted from V4.3 or V4.4 or V4.5 PL7 to V2.0 Unity Pro.

NOTE: These EFs are used as part of applications that implement Lexium controllers on a Fipio bus.

TSX WMY 100 Module

PL7 V4.3 service pack 1 is used in the TSX WMY 100 module. Applications containing this module are converted from PL7 ≥V4.3 service pack 1 to Unity Pro V2.0.

CANopen Applications

PL7 applications containing CANopen functionalities are converted to Unity Pro V2.0, according to the following rules:

- Only the TSX CPP 110 card is available using Unity Pro V2.0,
- A PL7 application containing a TSX CPP 110 card is entirely converted in Unity Pro V2.0,
- For PL7 applications containing a TSX CPP 100 card, the TSX CPP 100 card is replaced using Unity Pro with a TSX CPP 110 card, and you must check or modify the .CO configuration file of the CANopen bus for this to be implemented with a TSX CPP 110 card.

PL7 application conversion procedure

II

Subject of this Part

This part presents the different steps involved in converting a PL7 application or PL7 DFB into Unity Pro.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
2	Conversion of a PL7 application	27
3	Conversion of a PL7 DFB	37
4	Analysis of a PL7 application converted into Unity Pro	45

Conversion of a PL7 application

2

Subject of this Chapter

This chapter describes the procedure for converting a complete PL7 application into a Unity Pro application.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General	28
Procedure for Converting a PL7 Application into a Unity Pro Application	30
Results of the PL7 application conversion	32
Conversion with the Conversion Wizard	33
Topological Addresses of PL7 Micro in the Conversion Wizard	35

General

Introduction

Converting a PL7 application into a Unity Pro application requires the following:

- the application to be saved using PL7 ≥V4.3 software,
- a level **TSX ••• ••4** application processor,
- the application, as well as the sections, functional modules and DFBs it contains to be unprotected,
- the application source file to be exported and saved.

Update

All PL7 applications earlier than version 4.0 must be updated, before being converted into a Unity Pro application. To update a PL7 application, execute the following actions running PL7 ≥V4.3.

Step	Action
1	Choose the **Open** command from the **File** menu.
2	Select the hard disk and/or directory containing the file to be opened.
3	Select the file to be opened; the name of this file then appears in the **File Name** field.
4	Confirm with **Open**.
5	In the **Application Browser**, double-click on the **Configuration** directory.
6	Double-click on the **Hardware configuration** sub-directory. Result: The **Configuration** screen appears.
7	Select a version **TSX ••• ••3** processor that is compatible with your application from the drop-down menu in the top left-hand corner of the **Configuration** screen. **Result**: The **Change Processor** screen appears.
8	Confirm your selection with **OK**.
9	Select the **Save** command from the **File** menu.

Disabling the protection

To deactivate the protection of a PL7 application, execute the following actions running PL7 ≥V4.3:

Step	Action
1	Select the **Properties** command from the **Edit** menu.
2	Select the **Protection** tab.
3	In the **Application** field, uncheck the **Global application protection** checkbox.
4	In the **Sections** field, check **Protection deactivated**.
5	Confirm with **OK**. The confirmation is only effective once the password is entered.

⚠ CAUTION
UNEXPECTED APPLICATION BEHAVIOR - INCOMPLETE PROCEDURE If a PL7 application to be converted contains DFBs whose protection cannot be disabled *(see page 38)*, the converter converts neither the DFB declaration nor the calls from the DFB in the application. The networks containing the call from a protected DFB are not converted: an error message is displayed in the output window *(see page 49)*. In the conversion report file, the list of protected DFBs whose codes have not been able to be converted is provided. **Failure to follow these instructions can result in injury or equipment damage.**

Exporting the source file

To export a PL7 application, execute the following actions running PL7 ≥V4.3:

Step	Action
1	Select the **Export application** command from the **File** menu.
2	Select the disk and/or directory where the file must be stored.
3	Enter the file name in the **Name** field.
4	Confirm with **Save**.

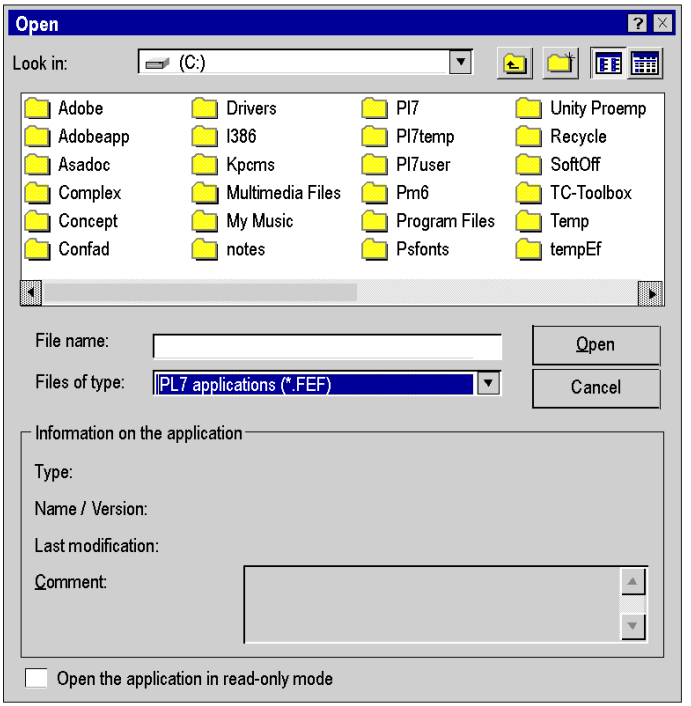
Procedure for Converting a PL7 Application into a Unity Pro Application

Introduction

The PL7 application converter is built into the Unity Pro software. The application to be converted is accessible using the command **File** →**Open**, which allows you to start the conversion.

Conversion Procedure

The following table describes the procedure for converting a PL7 application into a Unity Pro application.

Step	Action
1	Choose the **Open** command from the **File** menu.
2	In the **Files of type** field, select the type .fef (PL7 applications). 
3	Choose the hard drive and/or the directory containing the file to be converted.
4	Select the file (.fef) to be opened (and therefore converted). The name of this file then appears in the **File name** field.

Step	Action
5	Confirm with **Open**. **Result**: The conversion procedure is then started. The status bar shows how the procedure is progressing.
6	After the automatic import phase, you must start the analysis procedure *(see page 45)* manually in order to check the syntax of your application. **Note**: If during the import or analysis phase an output window *(see page 48)* is displayed on the screen, this means that there are conversion errors. In this case, correct these errors *(see page 49)*.

Results of the PL7 application conversion

Introduction

The results of the conversion of a PL7 application into a Unity Pro application are described in the following paragraphs.

Results

At the end of the conversion procedure, there are two possible scenarios:

- the application has been correctly converted,
- conversion errors have been generated.

If the application has been correctly converted, it is displayed on screen and may be saved in Unity Pro application format (.stu file).

In the event of conversion errors, you must manually correct the application in order for it to be exploitable. The output window *(see page 48)*, which can be used to manually correct these errors, is automatically displayed on the screen.

NOTE: the converted PL7 application may be saved in Unity Pro application format (.stu) even if conversion errors have not been corrected. If this is the case, the next time the application is opened, you must first launch an **Analysis** of the application, in order to display the output window on the screen *(see page 47)*.

NOTE: conversion errors appear in the report but not concerning errors not related to the program (e.g. an incomplete configuration in the FEF) are listed in a section located at the start of the master task.

Conversion with the Conversion Wizard

Introduction

The conversion wizard is an integrated part of Unity Pro.

You can use it to

- convert applications, exported out of legacy applications (PL7, Concept, LL984) to Unity Pro.
- convert legacy applications partially or as a whole.
- remap I/O objects (channels, variables etc.) during conversion by means of the wizard.
- adapt the hardware configuration of the new application concurrently in Unity Pro.
- modify the amount of used memory in the CPU.

The conversion wizard is available if you have chosen to install a converter (e.g. PL7 Application Converter) during the setup of Unity Pro.

Application Conversion as a Whole

To convert an application as a whole, while keeping the same PLC family and selection of application parts or remapping of I/O objects is not needed, you have to use the PL7 Application Converter directly via the Unity Pro menu **File** →**Open**.

NOTE: But use the **Conversion Wizard** to convert PL7 Micro applications to Unity Pro M340 applications (see next paragraph).

Partial Application Conversion

To convert an application partially and/or the PLC family must be changed or remapping of I/O objects is needed, use the conversion wizard via the Unity Pro menu **Tools** →**Convert Partially**.

For detailed information, please refer to the *Operating Modes Manual -> Conversion Wizard*.

General Procedure

General procedure to convert a legacy application to Unit Pro

Step	Action
1	Export your application out of your legacy programming system (e.g. as an FEF file out of PL7).
2	Create a new application in Unity Pro selecting a CPU with enough memory and the I/O access capabilities needed (e.g. Modicon M340). Optionally you can configure the I/O modules expected to be needed but you can modify the hardware configuration even later (see step 6).
3	Launch the conversion wizard in Unity Pro via **Tools** →**Convert Partially**. **Result:** The conversion wizard asks you to select the exported legacy source file.

Step	Action
4	Select the exported legacy source file. **Result:** The converter analyzes the source file and displays the result in the 3 tabs of the conversion wizard.
5	Select the parts of the application (or the complete application) to be converted in the **Structure** tab.
6	Remap the I/O objects for getting them compliant with the new hardware configuration. Concurrently you can modify the hardware configuration of the new application in Unity Pro. **Note:** To save a backup file of your intermediate I/O mapping you can use the **Save** button. With **Load** you can reload your latest saved intermediate I/O mapping.
7	After finishing all your selections and manual modifications click **OK**. **Result:** The converter applies the defined remapping to the selected parts of the source file and imports the results into the opened Unity Pro Application (e.g. an application with a Modicon M340 PLC).
8	Continue working on the opened application, save it or export as an XEF file.

Conversion Wizard Documentation

For detailed information on the conversion wizard, please refer to the *Operating Modes Manual -> Conversion Wizard*.

Topological Addresses of PL7 Micro in the Conversion Wizard

Overview

The **IO Remapping** tab of the conversion wizard shows the I/O objects of an analyzed legacy application in a table view.

By means of this table view you can remap I/O objects before conversion.

For detailed information on the conversion wizard, please refer to the *Conversion Wizard* section in the Operating Modes Manual *(see Unity Pro, Operating Modes)*.

Address Column

Topological addresses, coming from PL7 Micro are represented in the **Address** column of the conversion wizard with the following syntax:

```
%[MIQK]{[BWDF]}
{\<Bus Alias>.<Connection Point>\}
<Rack>.<Module>.
[[[<Channel>
 |<Sub Module>.<Sub Channel>]
  {.<Bit Selector>}]
 |MOD
 |ERR
 |MOD.ERR]
```

The above shown syntax is an extension of Unity Pro syntax to reflect the potential sub modules of PL7.

The different elements of the syntax are described below.

For detailed address description please refer to the *Direct Addressing Data Instances* section in the Languages Reference *(see Unity Pro, Program Languages and Structure, Reference Manual)*.

Brackets and separators

Element	Description
[...] square brackets	Square brackets imbed a choice.
| choice separator	A choice separator indicates a choice (OR).
{...} curly braces	Curly braces imbed an option.

Letters

Element	Description
M	Memory
I	input
Q	Output
K	constant
B	Byte
W	Word
D	Double Precision
F	Float

NOTE: Modicon M340 CPUs do not support %MD and %MF.

Element	Value	Description
Bus Alias	≥ 2	bus alias number for Unity Pro: ● 2 for FIPIO ● >2 for ASI and others
Connection Point	1 => 62 or any connection point	connection point of the module on the bus
Rack	0 => 7 or any rack number	● rack number for in-rack syntax ● rack number for FIPIO syntax ● 0 for ASI bus
Module	1 => 14 or any module number	● module number for in-rack syntax ● module number for FIPIO syntax ● 0 for ASI bus
Channel	0 => 63 or any channel number (MOD included) (PL7: 0..127 or MOD)	channel number (MOD excluded)
Sub Module	module dependent	for specific CPUs, analog modules, communication modules
Sub Channel	module dependent	for specific CPUs, analog modules, communication modules
Bit Selector	1 => 16	bit out of a word
MOD	module dependent	module information
ERR	module dependent	error bit
MOD.ERR	module dependent	module error bit

Syntax of I/O Objects

For the detailed syntax of I/O objects and their correspondences in Unity Pro please refer to *Correspondences between PL7 and Unity Pro: language objects, page 70*.

Conversion of a PL7 DFB

3

Subject of this Chapter

This chapter describes the procedure for converting a PL7 DFB into Unity Pro.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General	38
Procedure for converting a PL7 DFB into Unity Pro	41
Conversion of Protected DFBs	42
Procedure for importing a PL7 DFB into Unity Pro	43
Results of the PL7 DFB conversion	44

General

Introduction

Converting a PL7 DFB into Unity Pro requires the following:

- a PL7 ≥V4.3 application,
- the DFB to be unprotected,
- the DFB source file to be exported and saved.

Update

All PL7 applications earlier than version 4.3 must be updated to ≥V4.3 before being converted into a Unity Pro application *(see page 28)*.

Disabling the protection

To deactivate the protection of a PL7 DFB, execute the following actions running PL7 ≥V4.3:

Step	Action
1	In the application browser, double-click on the DFB type to be unprotected. **Result**: The **DFB type Editor** opens the screen for the selected DFB type.
2	Select the **Properties** command from the **Edit** menu.
3	Check the **Not protected** box.
4	Confirm with **OK**. **Note**: The confirmation is only effective once the password is entered.

⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR - CODE REWORK AFTER CONVERSION

The user's own diagnostics DFBs must be completed manually, after conversion.

To complete them:

- erase the private variables (`ADR_PROG` and `INST_NAME` for `UREGDFB`, `ADR_PROG`, `INST_NAME` and `COMMENT` for `REGDFB`),
- create two private variables `PIN_NB` (type INT) and `PIN_VAL` (type BOOL),
- modify the algorithm of the DFB in order to calculate the values of `PIN_NB` and `PIN_VAL` which contain the pin number in error and its expected value,
- modify the call parameters of the EF `REGDFB`: replace the parameters `ADR_PROG`, `COMMENT`, `INST_NAME` with `PIN_NB` and `PIN_VAL`,
- modify the call parameters of the EF `UREGDFB`: replace the parameters `ADR_PROG`, `INST_NAME` with 1, `PIN_NB` and `PIN_VAL`,
- adapt the DFB code.

Failure to follow these instructions can result in injury or equipment damage.

⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR - INCOMPLETE CONVERSION

The Schneider PL7 Diagnostics DFBs ALRM_DIA, EV_DIA, MV_DIA, NEPO_DIA, TEPO_DIA and SAFETY_MONITOR are automatically converted to Schneider Unity Pro Diagnostics DFBs. It is therefore not necessary to unprotect these DFBs before starting the conversion procedure.

The other PL7 diagnostics DFBs are not converted.

All Schneider Diagnostics DFBs available using Unity Pro are catalogued in the diagnostics family *(see Unity Pro, Diagnostics, Block Library)* of the diagnostics library.

Failure to follow these instructions can result in injury or equipment damage.

Exporting the source file

To export a PL7 DFB, execute the following actions running PL7 ≥V4.3:

Step	Action

1	• To export from the application browser:
	  • select the type of DFB by left-clicking.
	• To export from the DFB type editor:
	  • double-click on the DFB type,

2	Select the **Export** command from the **File** menu.
3	Select the disk and/or directory where the file must be stored.
4	Enter the file name in the **Name** field.
5	Confirm with **Save**.

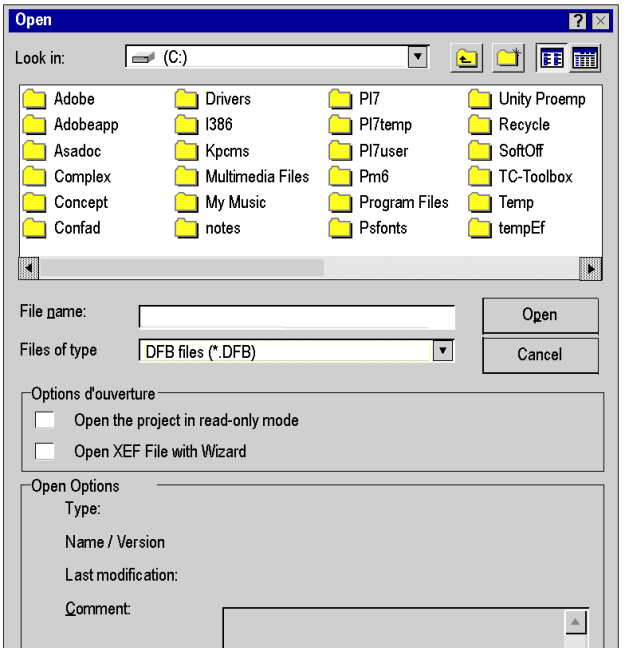
Procedure for converting a PL7 DFB into Unity Pro

Introduction

The PL7 application converter is built into the Unity Pro software. The DFB to be converted is accessible using the command **File →Open**, which allows you to start the conversion.

Conversion procedure

The following table describes the procedure for converting a PL7 DFB into Unity Pro.

Step	Action
1	Choose the **Open** command from the **File** menu.
2	In the **Files of type** field, select the type .dfb (DFB file). 
3	Choose the hard drive and/or the directory containing the file to be converted.
4	Select the file (.dfb) to be opened (and therefore converted). The name of the file then appears in the **File name** field.
5	Confirm with **Open**.
6	The conversion procedure is then started. The status bar shows how the procedure is progressing.
7	At the end of conversion, a screen appears indicating the end of the procedure.

Conversion of Protected DFBs

Procedure

If you have sold protected DFBs to your customers, you must use the following procedure to convert them from PL7 to Unity Pro.

Step	Action
1	In PL7 clear the code for your protected DFBs.
2	Unprotect your DFBs.
3	Send your customers the unprotected DFBs (without the code) and ask them to replace the protected DFBs in their PL7 applications by these unprotected DFBs.
4	Ask your customers to convert their applications from PL7 to Unity Pro.
5	Convert your DFBs (with the code) and protect them.
6	Send them to your customers.
7	Ask your customers to update the unprotected DFBs with the protected DFBs containing the code.

Procedure for importing a PL7 DFB into Unity Pro

Introduction

When a PL7 DFB is converted into Unity Pro, you must manually launch the import and analysis operations that follow conversion.

Import procedure

The following table describes the procedure for initializing the import and analysis of a PL7 DFB converted into Unity Pro.

Step	Action
1	Open a Unity Pro application.
2	Select the **Import** command from the **File** menu.
3	Select the hard disk and/or directory containing the file to be imported.
4	Select the file to be imported. The name of this file then appears in the **File name** field.
5	Confirm with **Open**, which launches the import procedure.
6	When the screen indicating the end of the import procedure is displayed, select the **Analyse** command from the **Build** menu, which launches the analysis procedure.
7	Should any conversion errors occur, the output window *(see page 48)* is displayed on the screen, allowing you to correct them.

Results of the PL7 DFB conversion

Introduction

The results of the conversion of a PL7 DFB into Unity Pro are described in the following paragraphs.

Results

At the end of the conversion procedure, there are two possible scenarios:

- the DFB has been correctly converted,
- conversion errors have been generated.

If the DFB has been correctly converted, it may be used in a Unity Pro application and saved in the application format (.stu file).

In the event of conversion errors, you must manually correct the DFB in order for it to be exploitable. The output window *(see page 48)*, which can be used to manually correct these errors, is automatically displayed on the screen at the end of the analysis procedure.

NOTE: The Unity Pro application containing the converted DFB may be saved in **.stu** format even if the DFB conversion errors have not been corrected. If this is the case, the next time the application is opened, you must first launch *(see page 47)*an **Analysis** of the application, in order to display the output window on the screen.

Analysis of a PL7 application converted into Unity Pro

4

Subject of this Chapter

This chapter presents the analysis phase of a PL7 application converted into Unity Pro.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General	46
Analysis Procedure	47
End of the analysis procedure	48
'Converter error' message in the analysis procedure	49
Miscellaneous errors in the analysis procedure	50

General

Introduction

The analysis allows the detection of errors generated in the application during conversion

Analysis

During the analysis phase, the following errors are detected:

- syntax errors,
- semantic errors,
- missing program parts,
- objects that have no Unity Pro equivalent,
- graphic objects or Grafcet drawings that have no Unity Pro equivalent,
- other errors (EFs developed by the user, etc.)

Output window

All types of error detected during the analysis phase are automatically shown on the screen in the output window.

Errors necessitating manual correction are signaled by the message `"Converror"`.

This message, which appears in inverted commas in the output window allows you to directly access the part of the program to be corrected by a left double click.

Analysis Procedure

Introduction

The analysis procedure must be manually launched after the automatic import phase.

Manual analysis procedure

The following table describes the procedure for manually running the analysis phase.

Step	Action
1	Select the **Analyse** command from the **Build** menu, which launches the analysis procedure.
2	Should any conversion errors occur, an output window is displayed on the screen, allowing you to correct them.

End of the analysis procedure

Introduction

The procedure of analyzing a PL7 application or a PL7 DFB converted into Unity Pro ends when the output window is displayed on the screen.

End of analysis

Once the analysis procedure is over, there are 2 possible scenarios:

- the output window contains `CONVERROR` messages: in order for the converted application or DFB to be exploitable, you must manually correct these conversion errors *(see page 49)* and delete the corrected `CONVERROR` messages before saving,
- the output window does not contain `CONVERROR` messages: the converted application or DFB is directly exploitable for compilation and transfer to the PLC.

NOTE: At any time during the manual correction of the converted PL7 application or PL7 DFB, you must delete the corrected `CONVERROR` messages and save the application in Unity Pro format (**.stu** file). If this is the case, the next time the application is opened, you must first launch an **Analysis** of the application, *(see page 47)*in order to display the output window on the screen.

’Converter error’ message in the analysis procedure

Introduction

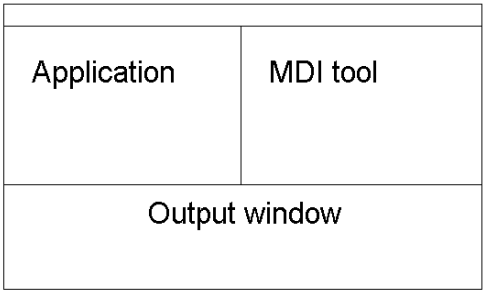
The `Converter error` message appears in the output window if:

- you have to correct the errors resulting from the conversion,
- you have to complete missing programming parts.

NOTE: You must correct the errors generated by the conversion then save application and analyse the converted project.

Illustration

The following diagram shows the Unity Pro software screen at the end of the procedure of analyzing a PL7 application or a converted PL7 DFB.



Description

The following table describes the various parts of the Unity Pro screen.

Part	Description
Application	Browser of the converted PL7 application or the Unity Pro application into which you are importing the converted PL7 DFB.
Output window	Window containing all conversion or analysis error messages.
`"Converter error"`	Message displayed if you have to manually correct part of the program of the converted application or DFB. By double-clicking on the left button of the mouse on the word `"Converter error"`, you access the MDI tool.
MDI tool	The part of the program to be corrected associated with the `"Converter error"` message is displayed in the field. You can correct the error directly in the MDI tool using Unity Pro commands.

Miscellaneous errors in the analysis procedure

Introduction

Conversion error messages are normally displayed in the output window.

The following paragraphs contain examples of these errors.

User-developed EFs

If the PL7 application to be converted contains EFs developed by the user with **TLX L SDKC PL7 40M** software, the converter does not recognize them and therefore cannot convert them.

A warning message is then displayed in the conversion report file and in the output window.

Availability of EFs

During the conversion of an EF, the converter does not guarantee the availability of the equivalent Unity Pro EF.

In the event that, after converting the EF from the PL7 library, the equivalent EF is missing from the Unity Pro library, a conversion error is displayed.

A warning message is then displayed in the conversion report file and in the output window.

Correspondence between PL7 and Unity Pro

III

Subject of this Part

This part contains the tables of correspondence between PL7 programming and its Unity Pro equivalent, as well as the equivalence tables for hardware devices (processor + memory card).

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
5	Inter-platform equivalence	53
6	Correspondence between application structures	59
7	Correspondences between common language elements	65
8	Correspondences between ladder language elements	137
9	Correspondences between Structured Text language elements	143
10	Correspondences between Instruction List language elements	147
11	Correspondences between Grafcet language elements	157
12	Other correspondences between PL7 and Unity Pro elements	161

Inter-platform equivalence

5

Subject of this Chapter

This chapter contains the hardware equivalence tables (processor + memory card).

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Premium processors	54
Atrium Processors	57

Premium processors

Introduction

The PL7 application converter ensures that the converted application is operational by providing a table of correspondences between the original processor and the updated processor.

Each processor is associated with an equivalent processor. If the original processor is equipped with a memory card (with or without data storage), the equivalent processor will also be equipped with the same type of card, except for the few specific cases indicated in the legend of the following table.

Processor Equivalence

The following table indicates the equivalence between original processors and updated processors.

Original processor	Processor: upgrade to be performed using PL7 V4	Updated processor	Action performed
TSX P57 10/102	**TSX P 57103** (1)	**TSX P57 104**	Processor replaced
T PMX P57 10/102	**TSX P 57203** (2)	**TSX P57 204**	Processor replaced
TSX P57 103 (1)	Not necessary	**TSX P57 104**	Processor replaced
TSX P57 153 (1)	Not necessary	**TSX P57 154**	Processor replaced
TSX P57 20/202	**TSX P 57203** (2)	**TSX P57 204**	Processor replaced
TSX P57 252	**TSX P 57253** (2)	**TSX P57 254**	Processor replaced
T PMX P57 202	**TSX P 57203**	**TSX P57 204**	Processor replaced
TSX P57 203 (1)	Not necessary	**TSX P57 203** with new OS	Processor version updated
TSX P57 253 (1)	Not necessary	**TSX P57 253** (2) with new OS	Processor version updated
TSX P57 2623 (1)	Not necessary	**TSX P57 2623** with a new OS, identified by Unity as a TSX P57 2634	Processor version updated
TSX P57 30/302	**TSX P 57303** (1) (2)	**TSX P57 304**	Processor replaced
TSX P57 352	**TSX P 57353** (1) (2)	**TSX P57 354**	Processor replaced
T PMX P57 352	**TSX P 57353** (1)	**TSX P57 354**	Processor replaced
TSX P57 303 (1)	Not necessary	**TSX P57 303** with new OS	Processor version updated
TSX P57 303A (1)	Not necessary	**TSX P57 303A** with new OS	Processor version updated

Original processor	Processor: upgrade to be performed using PL7 V4	Updated processor	Action performed
TSX P57 353 (1)	Not necessary	**TSX P57 353** with new OS	Processor version updated
TSX P57 353A (1)	Not necessary	**TSX P57 353A** with new OS	Processor version updated
TSX P57 3623 (1)	Not necessary	**TSX P57 3623** with a new OS, identified by Unity as a TSX P57 3634.	Processor version updated
TSX P57 3623A (1)	Not necessary	**TSX P57 3623A** with a new OS, identified by Unity as a TSX P57 3634.	Processor version updated
TSX P57 402	**TSX P 57453** (1)	**TSX P57 454**	Processor replaced
TSX P57 452	**TSX P 57453** (1)	**TSX P57 454**	Processor replaced
T PMX P57 452	**TSX P 57453** (1)	**TSX P57 454**	Processor replaced
TSX P57 453 (1)	Not necessary	**TSX P57 454**	Processor replaced
TSX P57 453A (1)	Not necessary	**TSX P57 454**	Processor replaced
TSX P57 4823A (1)	Not necessary	No equivalence	-
Legend:			
(1)	Processors that do not accept 160K-word cards. Using PL7, 57 1•• type processors do not accept memory cards greater than 64K words.		
(2)	These processors require two slot spaces, whereas the initial processors using PL7 require only one.		

NOTE: Processors **TSX P57 2823** and **TSX P57 4823** are not converted. You must therefore adapt your configuration using PL7 in order to convert it into Unity Pro.

Memory card equivalence

For all processors except those indicated (1) in the above table, the cartridge correspondence is as follows:

Memory card using PL7	Corresponding memory card using Unity Pro
None	None
32 K words	96 K bytes
64 K words	192 K bytes
128 K words	384 K bytes

Memory card using PL7	Corresponding memory card using Unity Pro
128 K words + Storage	768 K bytes
160 K words	448 K bytes
160 K words + Storage	1000 K bytes
256 K words	768 K bytes
256 K words + Storage	2000 K bytes
384 K words	2000 K bytes
512 K words + Storage	2000 K bytes

Atrium Processors

Introduction

The PL7 application converter ensures that the converted application is operational by providing a table of correspondence between the original processor and the updated processor.

Each processor is associated with an equivalent processor. If the original processor is equipped with a memory card (with or without data storage), the equivalent processor will also be equipped with the same type of card, except for the few specific cases indicated in the legend of the following table.

Processor Equivalence

The following table indicates the equivalence between original processors and updated processors.

Original processor	Processor: upgrade to be performed running PL7 V4	Updated processor	Action performed
T PCX 57 1012	**T PCX 57 203**	**TSX PCI 57 204**	Processor replaced
T PCX 57 203	Not necessary	**TSX PCI 57 204**	Processor replaced
T PCX 57 3512	**T PCX 57 353**	**TSX PCI 57 354** (2)	Processor replaced
T PCX 57 353 (1)	Not necessary	**TSX PCI 57 354** (2)	Processor replaced
Key:			
(1)	Processors that do not accept 160K-word cards.		
(2)	These processors are available for Unity Pro versions higher than V1.0.		

Memory Card Equivalence

For all processors except those indicated (1) in the above table, the cartridge correspondence is as follows:

Memory card using PL7	Corresponding memory card using Unity Pro
None	None
32 K words	96 K bytes
64 K words	192 K bytes
128 K words	384 K bytes
128 K words + Storage	768 K bytes
160 K words	448 K bytes

Memory card using PL7	Corresponding memory card using Unity Pro
160 K words + Storage	1,000 K bytes
256 K words	768 K bytes
256 K words + Storage	2,000 K bytes
384 K words	2,000 K bytes
512 K words + Storage	2,000 K bytes

Correspondence between application structures

6

Subject of this Chapter

This chapter contains the tables of correspondence between PL7 and Unity Pro application structures.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Correspondences between PL7 and Unity Pro: structural elements	60
6.2	Correspondences between PL7 and Unity Pro: functional modules	64

6.1 Correspondences between PL7 and Unity Pro: structural elements

Subject of this Section

This section contains the tables of correspondence between PL7 application structure elements and their Unity Pro equivalents.

PL7 application structure elements are divided into three categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),
- those that have no Unity Pro equivalent (status: deleted).

NOTE: if a PL7 element has no Unity Pro correspondence, a warning and a conversion error message are generated *(see page 46)*.

What's in this Section?

This section contains the following topics:

Topic	Page
Tasks, events, and SRs	61
The sections	62
LD Sections Split into Small Sections	63

Tasks, events, and SRs

Introduction

Conversion replaces PL7 tasks EVT and SR by their Unity Pro equivalents.

Tasks, EVTi, SR

The following table describes any correspondence and differences between the EVT and SR tasks in PL7 and Unity Pro.

	PL7	**Unity Pro**	**Status**
MAST task	Cyclic or periodic	Cyclic or periodic	Converted
FAST task	Periodic	Periodic	Converted
Event processing: EVTi	The number of events available depends on the processor	The number of events available depends on the processor	Converted
	The system words manage the events	The system words manage the events	Converted (1)
	`MASKEVT` `UNMASKEVT`	`MASKEVT` `UNMASKEVT`	Converted (2)
EVTi: priority level	Priority level management	Priority level management	Converted
Subroutines: SRi	SRi	SR Section	Modified (3)
Legend:			
(1)	The same system objects exist in Unity Pro *(see page 75)*.		
(2)	The same EFs exist in Unity Pro *(see page 120)*.		
(3)	The SR name is modified but the operation remains the same. In Unity Pro, the SRi becomes an SR section named SRi().		

The sections

Introduction

Conversion replaces PL7 section characteristics by their Unity Pro equivalents.

Sections

The following table describes any correspondence and differences between PL7 and Unity Pro section characteristics.

		PL7	Unity Pro	Status
Sections		Yes	Yes	Converted
Activation condition		Yes	Yes	Converted
Objects (1)		%Si %Mi %MWi:Xj %SWi:Xj %KWi:Xj %Mi[%MWj] %Mi[%SWj] %Mi[%KWj] ...	Equivalent Unity Pro objects	Modified (2)
Protection of sections		Write Read/Write None	Write Read/Write None	Converted
Section attributes				
Long name		16 characters	≥ 16 characters	Modified (3)
Short name		8 characters	≥ 8 characters	Modified
Comment		250 characters	256 characters	Modified
Number of sections in ...	MAST, FAST, AUXi	4096	No limitation	Modified
	EVT	1	1	Converted
	DFB	1	≥ 1	Modified
	SR	1	1	Converted
Language		LD, ST, IL	LD, ST, IL	Converted
Legend:				
(1)		Objects that define the execution condition of a section.		
(2)		The PL7 application converter replaces these objects by their Unity Pro equivalents *(see page 70)*.		
(3)		The section name cannot already be used to define one of the variables of the application.		

LD Sections Split into Small Sections

Introduction

In PL7 there is no size restriction for program sections.

Sections of 1000 rungs are existing and that leads to several drawbacks:

- Performances in edition (opening, inserting lines in a section) may take long time.
- Performance during online modifications may take some time, since the whole section is analyzed, compiled and transferred.
- Online modification of a big section may drive to switch back offline to perform the modification.

Splitting Feature

A splitting feature is implemented, which splits LD sections at a given amount of networks, if no jump scope is active.

Networks per Section

Before starting the converting process in Unity Pro the **Networks per Section** option can be set via **Tools** →**Options**. You can select a value of 1 to 30 networks per section.

Jump Scopes

A section may contain jumps to labels. A jump is located inside the section.

The converter takes jumps inside the section into account since it is not allowed to have a jump from one section to another. Depending on how the jump is validated inside the section (active or not), the automatic splitting at network bounds is inhibited (if active).

The feature of recognizing jump scopes can be suppressed by the **Ignore jumps when splitting** option. This option can be set via **Tools** →**Options** in Unity Pro. If this option is set, automatic section splitting always takes place regardless of jumps. Therefore many error messages concerning jump targets will be generated at analyze time.

NOTE: There are forward and backward jumps possible in PL7 applications.

Functional Modules

If during conversion LD sections are split into smaller sections, **Functional Modules** will be created in Unity Pro.

The functional modules are named as the original sections in PL7.

A functional module collects all parts of the original section.

6.2 Correspondences between PL7 and Unity Pro: functional modules

Functional modules

Introduction

Conversion replaces PL7 functional module characteristics by their Unity Pro equivalents.

Functional modules

The following table describes any correspondence and differences between PL7 and Unity Pro functional module characteristics.

			PL7	Unity Pro	Status
Functional modules			Yes	Yes	Converted
Nesting of functional modules			No limitation	No limitation	Converted
Comment			0..127 characters	0..255 characters	Modified
Size of description file			No limitation	No limitation	Converted
Maximum capacity of a functional module	Number of functional modules		No limitation	No limitation	Converted
	Number of sections	LD, ST, IL	No limitation	No limitation	Converted
		Grafcet	1	No limitation	Modified (1)
	Number of events in a section		No limitation	No limitation	Converted (2)
	Number of macro steps:		Limited by the processor	None	Deleted (1)
	Number of animation tables		No limitation	No limitation	Converted
	Number of runtime screens		No limitation	No limitation	Converted

Legend:	
(1)	The PL7 application converter does not convert all types of functional modules *(see page 165)*.
(2)	The possible programming languages are the following: ● LD, ST, IL

Correspondences between common language elements

7

Subject of this Chapter

This chapter contains the tables of correspondences between the objects, instructions and SFBs common to the different languages.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Correspondences between PL7 and Unity Pro: types and tables	66
7.2	Correspondences between PL7 and Unity Pro: language objects	70
7.3	Correspondences between PL7 and Unity Pro: instructions	95
7.4	Correspondences between PL7 and Unity Pro: SFBs	127

7.1 Correspondences between PL7 and Unity Pro: types and tables

Subject of this Section

This section contains the tables of correspondence between PL7 types and tables and their Unity Pro equivalents.

PL7 types and tables are divided into two categories :

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified).

What's in this Section?

This section contains the following topics:

Topic	Page
Types and arrays	67
Operations between mixed types	69

Types and arrays

Introduction

Conversion replaces the PL7 types and arrays by their Unity Pro equivalent.

Types

The following table describes any correspondence and differences between PL7 and Unity Pro types.

	PL7	Unity Pro	Status
Type	BOOL	BOOL	Converted
	EBOOL	EBOOL	Converted
	WORD	INT	Modified (1)
	DWORD	DINT	Modified (1)
	REAL	REAL	Converted
Legend:			
(1)	The WORD and DWORD types are converted into INT and DINT types *(see page 169)*.		

Arrays

The following table describes any correspondence and differences between PL7 and Unity Pro arrays.

	PL7	Unity Pro	Status
Table	Boolean array (EBOOL) `%Mi:n`	ARRAY [0..n-1] OF EBOOL	Modified (1)
	Word array (WORD) `%MWi:n`	ARRAY [0..n-1] OF EBOOL	Modified (1)
	Array of double words (DWORD) `%MDi:n`	ARRAY [0..n-1] OF DINT	Modified (1)
	Floating point array (REAL) `%MFi:n`	ARRAY [0..n-1] OF REAL	Modified (1)
	Bytes array `%MBi:n`	STRING [n]	Modified (1)
Legend:			
(1)	The converter modifies the declaration *(see page 169)*.		

Specific types and arrays

The following table describes any correspondence and differences between specific PL7 and Unity Pro types and arrays.

	PL7	Unity Pro	Status
Type	The TIME, DATE and TOD formats are double words (DWORD).	DINT	Modified (1)
Table	The DT format is an array of 4 words `%MWi:4`.	ARRAY[0..3] OF INT	Modified (1)
Legend:			
(1)	WORD and DWORD types are converted into INT and DINT types *(see page 169)*.		

Operations between mixed types

Introduction

Operations between mixed types are not always possible in Unity Pro. Conversion replaces these PL7 operations by their Unity Pro equivalent.

Mixed types

The following table describes any correspondence and differences between operations between mixed types.

Operations	PL7	Unity Pro	Status
Between words and double words (indexed or not) Between a word table and a double word table	Yes	No	Modified (1)
Between 2 word or double word tables	Yes	Yes	Converted (2)
Between word and double word tables and words or double words	Yes	No	Modified (3)
Assignment of a bit table into another bit table	Yes	Yes	Converted
Assignment of a word or double word table into a bit table	Yes	No	Modified (4)
Legend:			
(1)	Operations between words and double words are not possible in Unity Pro *(see page 170)*.		
(2)	This type of operations is possible in Unity Pro *(see page 109)*.		
(3)	Operations between word or double word tables and words or double words are not possible in Unity Pro *(see page 170)*.		
(4)	The assignment of a word or double word table into a bit table is not possible in Unity Pro *(see page 170)*.		

7.2 Correspondences between PL7 and Unity Pro: language objects

Subject of this Section

This section contains the tables of correspondences between PL7 language objects and their Unity Pro equivalent.

The PL7 objects are split into three categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),
- those that have no Unity Pro equivalent (status: deleted).

NOTE: if a PL7 object has no Unity Pro correspondence, a warning and a conversion error message are generated *(see page 46)*.

What's in this Section?

This section contains the following topics:

Topic	Page
Immediate values	71
Labels	74
System objects	75
Memory objects (variables and constants)	77
Word bits	80
Symbolized Directly Addressed Variables	81
Symbolized tables and indexed objects	82
SFBs	86
In-rack Input/Output objects	89
Remote input/output objects	91
Grafcet objects	93

Immediate values

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Immediate values

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro			Status
	Syntax	**Type of data**	**Syntax**		**Type of data**	
Boolean	`FALSE/TRUE`	BOOL	`0/1` or `FALSE/TRUE`		BOOL	Converted
Decimal integer (base 10)	`1234`	WORD	`1234`		INT	Converted
Long decimal integer (base 10)	`123456`	DWORD	`123456`		DINT	Converted
Base 2 integer	`2#x...` (1...16 figures)	WORD	If `2#x ...` ≤+32767	`2#x...`	INT	Modified
			If `2#x ...` > +32767	decimal value (1)		
Base 2 long integer	`2#x...` (17...32 figures)	DWORD	If `2#x ...` ≤+2147483647	`2#x...`	DINT	Modified
			If `2#x ...` > +2147483647	decimal value (1)		
Base 16 integer	`16#x...` (1...4 figures)	WORD	If `16#x ...` ≤+32767	`2#x...`	INT	Modified
			If `16#x ...` > +32767	decimal value (1)		
Base 16 long integer	`16#x...` (5...8 figures)	DWORD	If `16#x ...` ≤+2147483647	`2#x...`	DINT	Modified
			If `16#x ...` > +2147483647	decimal value (1)		
Real	`-1.32e-12`	REAL	`-1.32e-12`		REAL	Converted
Character string	`'aAbBcC'`	STRING	`'aAbBcC'`		STRING	Converted
Network address	`ADR#...` (2)	AR_W:6	`ADDR` (2)		ARRAY OF INT	Modified

Legend:	
(1)	This decimal value may be negative if the sign bit is equal to 1. *(see page 172)*
(2)	See table below.

Network address

The following table describes any correspondence and differences between PL7 and Unity Pro network address objects.

PL7		Unity Pro	
Syntax	**Type of data**	**Syntax (1)**	**Type of data**
ADR#xy.i.e ADR#2.1.3	AR_W:6	ADDR('r.m.c.d') ADDR('0.2.1.3')	ARRAY OF INT
ADR#xy.i.SYS ADR#102.1.SYS	AR_W:6	ADDR('r.m.c.SYS') ADDR('1.2.1.SYS')	ARRAY OF INT
ADR#xy.SYS ADR#102.SYS	AR_W:6	ADDR('r.m.SYS') ADDR('1.2.SYS')	ARRAY OF INT
ADR#SYS	AR_W:6	ADDR('SYS')	ARRAY OF INT
ADR#APP	AR_W:6	ADDR('APP')	ARRAY OF INT
ADR#\xy.i.c\SYS ADR#\4.0.23\SYS	AR_W:6	ADDR('\b.e\SYS') ADDR('\4.23\SYS') (2)	ARRAY OF INT
ADR#{r.s}xy.i.e ADR#{2.4}2.1.3	AR_W:6	ADDR('{n.s}r.m.c.d') ADDR('{2.4}0.2.1.3')	ARRAY OF INT
ADR#{r.s}xy.i.SYS ADR#{2.4}102.1.SYS	AR_W:6	ADDR('{n.s}r.m.c.SYS') ADDR('{2.4}1.2.1.SYS')	ARRAY OF INT
ADR#{r.s}xy.SYS ADR#{2.4}102.SYS	AR_W:6	ADDR('{n.s}r.m.SYS') ADDR('{2.4}1.2.SYS')	ARRAY OF INT
ADR#{r.s}SYS ADR#{2.4}SYS	AR_W:6	ADDR('{n.s}SYS') ADDR('{2.4}SYS')	ARRAY OF INT
ADR#{r.s}APP ADR#{4}APP	AR_W:6	ADDR('{n.s}APP') ADDR('{4}APP')	ARRAY OF INT
ADR#{r.s}APP.num, ADR#{2.4}APP.0	AR_W:6	ADDR('{n.s}APP.num'), ADDR('{2.4}APP.0')	ARRAY OF INT
ADR#{r.s}\xy.i.c\SYS ADR#{2.4}\4.0.23\SYS	AR_W:6	ADDR('{n.s}\b.e\SYS') ADDR('{2.4}\4.23\SYS') (2)	ARRAY OF INT

Legend:		
(1)		The address ADR# is replaced by an EF *(see page 172)*.
(2)		The converter selects the bus number.
PL7	**Unity Pro**	
x	r	Rack.
y	m	Position in the rack (module).

PL7			Unity Pro	
Syntax		**Type of data**	**Syntax (1)**	**Type of data**
i	c	Channel.		
e	d	Data. Operational if equal to zero.		
c	e	Connection point (equipment).		
r	n	Network. Operational if equal to zero.		
s	s	Station.		
num	num	SFB.		
-	b	Bus.		

Labels

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Labels

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Program label	`%Li` (1)	label	`Li`	label	Modified
DFB label	`Identifier`	label	`Identifier`	label	Converted
Legend:					
(1)	i = 0...999.				

System objects

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

System objects

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
System bit	%Si (1)	EBOOL	%Si or %SXi	BOOL	Converted
System word	%SWi (1)	WORD	%SWi	INT	Converted
Real-time clock system word	%SW50:4	AR_W	%SW50:4	ARRAY [0..3] OF INT at %SW50	Converted
System double word	%SDi	DWORD	%SDi	DINT	Converted
Legend:					
(1)	See table below.				

System bits and words

The following table describes any correspondence and differences between PL7 and Unity Pro bits and system words.

	PL7	Unity Pro	Status
	Syntax	**Syntax**	
System bits	%S36	-	Deleted
	%S37	-	
	%S95	-	
System words	%SW160	%SW76	Modified
	%SW161	%SW77	
	%SW162	%SW78	

	PL7	Unity Pro	Status
	Syntax	**Syntax**	
Grafcet system bits	%S21	SG7_21	Modified (1)
	%S22	SG7_22	
	%S23	SG7_23	
	%S24	SG7_24	
	%S25	-	Deleted
	%S26	-	
Grafcet system words	%SW20	-	Deleted
	%SW21	-	
	%SW22	SWG7_22	Modified (1)
	%SW23	SWG7_23	
	%SW24	SWG7_24	
	%SW25	SWG7_25	
	%SW125	-	Deleted
	%SW126	-	
	%SW127	-	
Legend:			
(1)	Grafcet system bits and words are replaced in Unity Pro by equivalent EFs *(see page 157)*.		

NOTE: all other system bits and words are converted.

Memory objects (variables and constants)

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Memory objects

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Internal bits	%Mi	EBOOL	%Mi or %Mxi	EBOOL	Converted
	%Mi[%MWj]	EBOOL	%Mi[%MWj]	EBOOL	
	%Mi[%KWj]	EBOOL	%Mi[%KWj]	EBOOL	
	%Mi[n]	EBOOL	%Mi[n]	EBOOL	
	%Mi:L	AR_X	%Mi:L	ARRAY OF EBOOL	
Internal words	%MWi	WORD	%MWi	INT	Converted
	%MWi[%MWj]	WORD	%MWi[%MWj]	INT	
	%MWi[%KWj]	WORD	%MWi[%KWj]	INT	
	%MWi[n]	WORD	%MWi[n]	INT	
	%MWi:L	AR_W	%MWi:L	ARRAY OF INT	
	%MWi[%MWj]:L	AR_W	%MWi[%MWj]:L	ARRAY OF INT	
	%MWi[%KWj]:L	AR_W	%MWi[%KWj]:L	ARRAY OF INT	
	%MWi[n]:L	AR_W	%MWi[n]:L	ARRAY OF INT	
Internal double words	%MDi	DWORD	%MDi	DINT	Converted
	%MDi[%MWj]	DWORD	%MDi[%MWj]	DINT	
	%MDi[%KWj]	DWORD	%MDi[%KWj]	DINT	
	%MDi[n]	DWORD	%MDi[n]	DINT	
	%MDi:L	AR_D	%MDi:L	ARRAY OF DINT	
	%MDi[%MWj]:L	AR_D	%MDi[%MWj]:L	ARRAY OF DINT	
	%MDi[%KWj]:L	AR_D	%MDi[%KWj]:L	ARRAY OF DINT	
	%MDi[n]:L	AR_D	%MDi[n]:L	ARRAY OF DINT	

		PL7		Unity Pro		Status
		Syntax	**Type of data**	**Syntax**	**Type of data**	
Internal reals		%MFi	REAL	%MFi	REAL	Converted
		%MFi[%MWj]	REAL	%MFi[%MWj]	REAL	
		%MFi[%KWj]	REAL	%MFi[%KWj]	REAL	
		%MFi[n]	REAL	%MFi[n]	REAL	
		%MFi:L	AR_R	%MFi:L	ARRAY OF REAL	
Constant words		%KWi	WORD	%KWi	INT	Converted
		%KWi[%MWj]	WORD	%KWi[%MWj]	INT	
		%KWi[%KWj]	WORD	%KWi[%KWj]	INT	
		%KWi[n]	WORD	%KWi[n]	INT	
		%KWi:L	AR_W	%KWi:L	ARRAY OF INT	
		%KWi[%MWj]:L	AR_W	%KWi[%MWj]:L	ARRAY OF INT	
		%KWi[%KWj]:L	AR_W	%KWi[%KWj]:L	ARRAY OF INT	
		%KWi[n]:L	AR_W	%KWi[n]:L	ARRAY OF INT	
Constant double words		%KDi	DWORD	%KDi	DINT	Converted
		%KDi[%MWj]	DWORD	%KDi[%MWj]	DINT	
		%KDi[%KWj]	DWORD	%KDi[%KWj]	DINT	
		%KDi[n]	DWORD	%KDi[n]	DINT	
		%KDi:L	AR_D	%KDi:L	ARRAY OF DINT	
		%KDi[%MWj]:L	AR_D	%KDi[%MWj]:L	ARRAY OF DINT	
		%KDi[%KWj]:L	AR_D	%KDi[%KWj]:L	ARRAY OF DINT	
		%KDi[n]:L	AR_D	%KDi[n]:L	ARRAY OF DINT	
Constant reals		%KFi	REAL	%KFi	REAL	Converted
		%KFi[%MWj]	REAL	%KFi[%MWj]	REAL	
		%KFi[%KWj]	REAL	%KFi[%KWj]	REAL	
		%KFi[n]	REAL	%KFi[n]	REAL	
		%KFi:L	AR_R	%KFi:L	ARRAY OF REAL	
Common words	network No. 0	%NW{j}k (1)	WORD	%NWs.d	INT	Modified
	other networks	%NW{i.j}k (1)	WORD	%NWn.s.d	INT	Modified
Variable-type character string		%MBi:L	STRING (2)	xxx_L:String[L]	STRING	Modified

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Constant-type character string	`%KBi:L`	STRING (2)	`xxx_L:String[L]`	STRING	Modified
Legend:					
(1)	**PL7**	**Unity Pro**			

	`i`
	`j`
	`k`
		`n`
		`s`
		`d`
			network number.
			station number.
			word number.		

(2)	Objects `%MBi:L` and `%KBi:L` are replaced by a character string *(see page 173)*.				

Word bits

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Word bits

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Bit j j = 1..15	%SWi:Xj	BOOL	%SWi.j	BOOL	Converted (1)
	%MWi:Xj	BOOL	%MWi.j	BOOL	
	%MWi[%MWj]:Xj	BOOL	%MWi[%MWj].j	BOOL	
	%MWi[%KWj]:Xj	BOOL	%MWi[%KWj].j	BOOL	
	%MWi[n]:Xj	BOOL	%MWi[n].j	BOOL	
	%KWi:Xj	BOOL	%KWi.j	BOOL	
	%KWi[%MWj]:Xj	BOOL	%KWi[%MWj].j	BOOL	
	%KWi[%KWj]:Xj	BOOL	%KWi[%KWj].j	BOOL	
	%KWi[n]:Xj	BOOL	%KWi[n].j	BOOL	
	%NW{r.s}k:Xj	BOOL	%NWn.s.d.j	BOOL	
	%NW{s}k:Xj	BOOL	%NWs.d.j	BOOL	
@ **= input/output addresses** (2)	%IW@:Xj	BOOL	%IW@.j	BOOL	Modified
	%QW@:Xj	BOOL	%QW@.j	BOOL	
	%MW@:Xj	BOOL	%MW@.j	BOOL	
	%KW@:Xj	BOOL	%KW@.j	BOOL	
Legend:					
(1)	The syntax of the bits extracted into Unity Pro is modified. *(see page 174)*				
(2)	The address @ has been replaced by the PL7 converter: ● conversion of local addresses *(see page 89)*, ● conversion of remote addresses *(see page 91)*.				

Symbolized Directly Addressed Variables

Introduction

In PL7 applications, especially for TSX Micro, direct addresses without symbols are frequently used.

Direct addressing in programs has the following drawbacks:

- These variables are not listed in the **Data Editor** in Unity Pro, because they do not have a symbol.
- To reuse parts of the program later is difficult, because the program cannot be changed easily with direct addresses.
- With addresses in the program, the programmer cannot switch located variables to unlocated variables.

For all these reasons, the converter provides a service to assign a symbol to direct addresses used in the program automatically.

Unique Symbol

A unique symbol is derived from the direct addresses by exchanging all non-letters and non-number characters into the underscore character.

PL7	Unity Pro
%MW100	MW100
%I0.2.3.4.	I0_2_3_4

Symbols Instead of Direct Addresses

Before starting the converting process in Unity Pro the **Symbols Instead of Direct Addresses** option can be set via **Tools** →**Options**.

Symbolized tables and indexed objects

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Tables of memory and constant objects

The following table describes any correspondence and differences between the symbolized tables of PL7 and Unity Pro memory and constant objects.

	PL7		Unity Pro		Status
	Address	**Associated symbol**	**Address**	**Associated variable (1)**	
Tables of memory and constant objects	%MWi:L	TABA:L	%MWi:L	TABA_L	Modified (2)
	%MDi:L	TABB:L	%MDi:L	TABB_L	
	%Mi:L	TABC:L	%Mi:L	TABC_L	
	%MFi:L	TABD:L	%MFi:L	TABD_L	
	%KWi:L	TABE:L	%MWi:L	TABE_L	
	%KDi:L	TABF:L	%MDi:L	TABF_L	
	%KFi:L	TABG:L	%MFi:L	TABG_L	
Legend:					
(1)	In Unity Pro, a variable is associated with each symbolized table *(see page 175)*.				
(2)	The table is declared from [0..L-1], its type and location depending on the address. **Example**: %MWi:L, declared in PL7 as TABA:L, in converted into TABA_L. TABA_L is therefore a declared table from [0..L-1], of type INT, located at %MWi and with the same comment as TABA.				

I/O object tables

The following table describes any correspondence and differences between the symbolized tables of PL7 and Unity Pro input/output objects.

	PL7		Unity Pro		Status
	Address	**Associated symbol**	**Address**	**Associated variable (1)**	
I/O object tables	%I@:L	TABA:L	%I@:L	TABA_L	Modified (2)
	%IW@:L	TABB:L	%IW@:L	TABB_L	
	%Q@:L	TABC:L	%Q@:L	TABC_L	
	%QW@:L	TABD:L	%QW@:L	TABD_L	
Legend:					
@	Addressing of the in-rack *(see page 89)* and/or remote *(see page 91)* input/output objects.				
(1)	In Unity Pro, a variable is associated with each symbolized table *(see page 175)*.				
(2)	The table is declared from [0..L-1], its type and location depending on the address. **Example**: %I@:L, declared in PL7 as TABA:L, in converted into TABA_L. TABA_L is therefore a declared table from [0..L-1], of type EBOOL, located at %I@ and with the same comment as TABA.				

Indexed memory and constant objects

The following table describes any correspondence and differences between PL7 and Unity Pro symbolized indexed memory and constant objects.

	PL7		Unity Pro		Status
	Address	**Associated symbol**	**Address**	**Associated variable (1)**	
Indexed memory and constant objects	%MWi[j]	TABA[j]	%MWi[j]	TABA_AR[j]	Modified (2)
	%MDi[j]	TABB[j]	%MDi[j]	TABB_AR[j]	
	%Mi[j]	TABC[j]	%Mi[j]	TABC_AR[j]	
	%MFi[j]	TABD[j]	%MFi[j]	TABD_AR[j]	
	%KWi[j]	TABE[j]	%KWi[j]	TABE_AR[j]	
	%KDi[j]	TABF[j]	%KDi[j]	TABF_AR[j]	
	%KFi[j]	TABG[j]	%KFi[j]	TABG_AR[j]	
Legend:					
(1)	In Unity Pro, a variable is associated with each indexed symbolized object *(see page 175)*.				
(2)	The table is declared from [0..NbMaxMW-i-1], its type and location depending on the address. **Example**: %MWi[j], declared in PL7 as TABA[j], in converted into TABA_AR[j]. TABA_AR[j] is therefore a declared table from [0..NbMaxMW-i-1], of type INT, located at %MWi and with the same comment as TABA.				

Indexed I/O objects

The following table describes any correspondence and differences between PL7 and Unity Pro symbolized indexed input/output objects.

	PL7		Unity Pro		Status
	Address	**Associated symbol**	**Address (1)**	**Associated variable**	
Indexed I/O objects	%I@[j]	TABA[j]	%I@[j]	-	Modified
	%IW@[j]	TABB[j]	%IW@[j]	-	
	%Q@[j]	TABC[j]	%Q@[j]	-	
	%QW@[j]	TABD[j]	%QW@[j]	-	
Legend:					
@	Addressing of the in-rack *(see page 89)* and/or remote *(see page 91)* input/output objects.				
(1)	Indexed input/output objects are converted into their non-symbolized form (address) *(see page 175)*.				

SFBs

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Premium Targets

For Premium, the SFBs are replaced with EFBs.

The EFBs have the same layout as the SFBs in PL7.

These EFBs are used in the converted applications.

Modicon M340 Targets

For Modicon M340, the SFBs are replaced with DFBs.

The DFBs have the same layout as the SFBs in PL7.

They include also some program in ST to adapt the interface to the IEC function blocks that are used in Unity Pro.

The DRUM SFB is represented by an empty DFB.

Quantum Targets

The conversion of SFBs is not available for Quantum.

SFB

The following table describes the converted PL7 SFBs.

SFB	Syntax	Type of data
PL7_3 Timer (3)	%Ti	T
enable input	%Ti.E	BOOL
control input	%Ti.C	BOOL
current value word	%Ti.ET	INT
preset value word	%Ti.PT	INT
time base	%Ti.TB	UINT
elapsed timer bit (Done)	%Ti.D	BOOL
current timer bit (Run)	%Ti.R	BOOL
PL7 Timers (3)	%TMi	TM
timer start input	%TMi.S	BOOL
current value word	%TMi.ET	INT
preset value word	%TMi.PT	INT

SFB	Syntax	Type of data
time base	%TMi.TB	UINT
current timer bit	%TMi.Q	BOOL
Monostable (3)	%MNi	M
start input	%MNi.S	BOOL
current value word	%MNi.ET	INT
preset value word	%MNi.PT	INT
time base	%MNi.TB	UINT
current timer bit (Run)	%MNi.R	BOOL
Up/down counter	%Ci	C
reset input	%Ci.R	BOOL
preset input	%Ci.P	BOOL
up-count input	%Ci.CU	BOOL
down-count input	%Ci.CD	BOOL
current value word	%Ci.CV	INT
preset value word	%Ci.PV	INT
downcounting overrun bit (Empty)	%Ci.E	BOOL
standby preset bit (Done)	%Ci.D	BOOL
upcounting overrun bit (Full)	%Ci.F	BOOL
Register	%Ri	R
reset input	%Ri.R	BOOL
store input	%Ri.I	BOOL
destock input	%Ri.O	BOOL
input word	%Ri.INW	INT
output word	%Ri.OUTW	INT
full register bit	%Ri.F	BOOL
empty register bit	%Ri.E	BOOL
register length	%Ri.LEN	UINT
register type	%Ri.FIFO	BOOL
Drum	%DRi	DR
reset input	%DRi.R	BOOL
up input	%DRi.U	BOOL
full drum bit	%DRi.F	BOOL
current step	%DRi.S	INT
duration word	%DRi.ET	INT

SFB	Syntax	Type of data
time base	%DRi.TB	UINT
number of steps	%DRi.LEN	UINT
i step states	%DRi.Wj j = 0..15	INT

In-rack Input/Output objects

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

In-rack I/O objects.

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Channel	%CHxy.0 %CH3.2	CHANNEL	%CHr.m.c %CH0.3.2	IODDT	Modified
Inputs %I					
Module or channel fault	%Ixy.i.ERR %I2.3.ERR	BOOL	%Ir.m.c.ERR %I0.2.3.ERR	BOOL	Modified
Bit	%Ixy.i.r %I2.3.1	EBOOL	%Ir.m.c.d %I0.2.3.1	EBOOL	Modified
	%Ixy.i.r[index] %I2.3.1[index]	EBOOL	%Ir.m.c.d[index] %I0.2.3.1[index]	EBOOL	
	%Ixy.i.r:L %I2.3.1:L	AR_X	%Ir.m.c.d:L %I0.2.3.1:L	ARRAY OF EBOOL	
Word	%IWxy.i.r %IW2.3.1	WORD	%IWr.m.c.d %IW0.2.3.1	INT	Modified
Double word	%IDxy.i.r %ID2.3.1	DWORD	%IDr.m.c.d %ID0.2.3.1	DINT	Modified
Real	%IFxy.i.r %IF2.3.1	REAL	%IFr.m.c.d %IF0.2.3.1	REAL	Modified
Outputs %Q					
Bit	%Qxy.i.r %Q2.3.1	EBOOL	%Qr.m.c.d %Q0.2.3.1	EBOOL	Modified
	%Qxy.i.r[index] %Q2.3.1[index]	EBOOL	%Qr.m.c.d[index] %Q0.2.3.1[index]	EBOOL	
	%Qxy.i.r:L %Q2.3.1:L	AR_X	%Qr.m.c.d:L %Q0.2.3.1:L	ARRAY OF EBOOL	
Word	%QWxy.i.r %QW2.3.1	WORD	%QWr.m.c.d %QW0.2.3.1	INT	Modified

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Double word	%QDxy.i.r %QD2.3.1	DWORD	%QDr.m.c.d %QD0.2.3.1	DINT	Modified
Real	%QFxy.i.r %QF2.3.1	REAL	%QFr.m.c.d %QF0.2.3.1	REAL	Modified
Variables %M					
Word	%MWxy.i.r %MW2.3.1	WORD	%MWr.m.c.d %MW0.2.3.1	INT	Modified
Double word	%MDxy.i.r %MD2.3.1	DWORD	%MDr.m.c.d %MD0.2.3.1	DINT	Modified
Real	%MFxy.i.r %MF2.3.1	REAL	%MFr.m.c.d %MF0.2.3.1	REAL	Modified
Constants %K					
Word	%KWxy.i.r %KW2.3.1	WORD	%KWr.m.c.d %KW0.2.3.1	INT	Modified
Double word	%KDxy.i.r %KD2.3.1	DWORD	%KDr.m.c.d %KD0.2.3.1	DINT	Modified
Real	%KFxy.i.r %KF2.3.1	REAL	%KFr.m.c.d %KF0.2.3.1	REAL	Modified
Character string	%KBxy.i.r:L %KF2.3.4:L	STRING	(1)	STRING	Modified

Legend:

(1)	The %KBxy.i.r:L and %KF2.3.4:L character strings are replaced by a single character string *(see page 173)*.

PL7	Unity Pro	
x	r	Rack.
y	m	Position in the rack (module).
i	c	Channel number.
r	d	Rank. Optional if equal to zero.

Remote input/output objects

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Remote Fipio bus objects

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7	Unity Pro	Status
	Syntax	**Syntax**	
Remote Fipio bus objects	%I\p.2.c\xy.i.r %I\0.2.34\1.2.1	%I\b.e\r.m.c.d %I\2.34\0.1.2.1 (1)	Modified

Legend:

(1) In Unity Pro, the bus identifier has been added. In order to convert Remote Fipio bus objects, the selected b identifier is assigned the value 2.

PL7	Unity Pro	
p	-	Module address.
c	e	Connection point.
i	c	Channel number.
r	d	Rank.
-	b	Bus identifier.
x	r	Rack.
y	m	Position in the rack.

Remote AS-i bus objects

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7	Unity Pro	Status
	Syntax	**Syntax**	
Remote AS-i bus objects	%I\xy.0\n.i %I\104.0\2.3	%I\b.e\r.m.c %I\15.2\0.0.3 (1)	Modified

Legend:

(1) In Unity Pro, the bus identifier has been added. In order to convert remote AS-i bus objects, the selected b identifier corresponds to the number of the xy rack/module.

PL7	Unity Pro	
x	r	Rack.
y	m	Position in the rack.
0	-	AS-i channel (the module **TSX SAY 100** has only one channel).
n	e	Slave number.
i	c	Channel.
-	b	Bus identifier.

Grafcet objects

Introduction

Conversion replaces the PL7 objects by their Unity Pro equivalent.

Grafcet objects

The following table describes any correspondence and differences between PL7 and Unity Pro objects.

	PL7		Unity Pro		Status
	Syntax	**Type of data**	**Syntax**	**Type of data**	
Grafcet objects	%Xi	BOOL	X_i.x	SFCSTEP_STATE	Modified (1)
	%XMj		XM_j.x		
	%Xj.i		X_j_i.x		
	%Xj.IN		X_j_IN.x		
	%Xj.OUT		X_j_OUT.x		
Grafcet objects	%Xi.T	WORD	X_i.t	SFCSTEP_STATE	Modified (1)
	%XMj.T		XM_j.t		
	%Xj.i.T		XM_j_i.t		
	%Xj.IN.T		XM_j_IN.t		
	%Xj.OUT.T		XM_j_OUT.t		
Legend:					
(1)	PL7 steps and macrosteps become, in Unity Pro, structured objects SFCSTEP_STATE. For the activity times of the steps, we advise you to check your program in order to use the TIME type instead of the a whole value, converted into PL7.				

Grafcet words and bits

The following table describes the correspondence and possible differences between Grafcet words and bits.

	PL7	Unity Pro	Status
	Syntax	**Syntax**	
Indexed Grafcet words and bits	%Xi[%MWj]	%Mi[%MWj]	Modified (1)
	%Xi.T[%MWj]	-	Deleted (2)
Grafcet words and bits table	%Xi:L	%Mi:L	Modified (1)
	%Xi.T:L	-	Deleted (2)
Legend:			
(1)	The objects of steps are no longer contiguous in the memory. Therefore, a set of %Mi objects is assigned by the PL7 application converter to simulate indexed Grafcet words or bits and the tables of Grafcet words or bits.		
(2)	You must replace these objects manually in the program.		

7.3 Correspondences between PL7 and Unity Pro: instructions

Subject of this Section

This section contains the tables of correspondences between PL7 common language instructions and their Unity Pro equivalent.

The PL7 instructions are split into three categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),
- those that have no Unity Pro equivalent (status: deleted).

NOTE: if a PL7 instruction has no Unity Pro correspondence, a warning and a conversion error message are generated *(see page 46)*.

What's in this Section?

This section contains the following topics:

Topic	Page
Boolean instructions	97
Comparison instructions	98
Bit table instructions	100
Arithmetic instructions	101
Logic instructions	104
Shift instructions	105
Numerical conversion instructions	107
Table instructions	109
Character String Instructions	115
Time management instructions	116
Exchange Instructions	117
Input/output instructions	118
Process control instructions	119
Other instructions	120
Communication instructions	121

Topic	Page
TCP Open instructions	122
Diagnostics instructions	124
Grafcet instructions	125
Human Machine Interface (HMI) instructions	126

Boolean instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
Operator	:=	:=	Converted
	AND	AND	Converted
	OR	OR	Converted
	XOR	XOR	Converted
	NOT	NOT	Converted
	RE	RE	Modified (1)
	FE	FE	Modified (1)
	SET	SET	Modified (1)
	RESET	RESET	Modified (1)
Legend:			
(1)	Replaced by an EF.		

Comparison instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD)	**Unity Pro (INT)**	**Status**
Operator	>	>	Converted
	<	<	Converted
	≤	≤	Converted
	≥	≥	Converted
	=	=	Converted
	≠	≠	Converted

Long integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (DWORD)	**Unity Pro (DINT)**	**Status**
Operator	>	>	Converted
	<	<	Converted
	≤	≤	Converted
	≥	≥	Converted
	=	=	Converted
	≠	≠	Converted

Real words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (Floating point)	**Unity Pro (REAL)**	**Status**
Operator	>	>	Converted
	<	<	Converted
	≤	≤	Converted
	≥	≥	Converted
	=	=	Converted
	≠	≠	Converted

Bit table instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
Operator: between two bit tables	:=	:=	Converted
Operator: between a bit table and an integer or long integer word	:=	MOVE_INT_AREBOOL MOVE_DINT_AREBOOL	Modified (1)
Operator: between an integer or long integer word and a bit table	:=	MOVE_AREBOOL_INT MOVE_AREBOOL_DINT	Modified (1)
EF	BIT_D	COPY_AREBOOL_ARDINT	Modified (2)
	BIT_W	COPY_AREBOOL_ARINT	Modified (2)
	COPY_BIT	COPY_AREBOOL_AREBOOL	Modified (2)
	D_BIT	COPY_ARDINT_AREBOOL	Modified (2)
	LENGTH_ARX	LENGTH_AREBOOL	Modified (3)
	W_BIT	COPY_ARINT_AREBOOL	Modified (2)
Legend:			
(1)	The operator is replaced by an EF, which depends on the type of operand.		
(2)	The function name is modified, but the operation remains the same except in the case of negative ranks *(see page 178)*.		
(3)	The function name is modified but the operation remains the same.		

Arithmetic instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD)	**Unity Pro (INT)**	**Status**
Operator	+	+	Converted
	-	-	Converted
	*	*	Converted
	/	/	Converted
	REM	MOD	Modified (1)
EF	SQRT	SQRT_INT	Modified (2)
	ABS	ABS_INT	Modified (2)
Operator	INC	INC_INT	Modified (2)
	DEC	DEC_INT	Modified (2)
	:=	:=	Converted
Legend:			
(1)	The operator name is modified but the operation remains the same.		
(2)	Replaced by an EF whose name depends on the type of data processed.		

Long integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (DWORD)	**Unity Pro (DINT)**	**Status**
Operator	+	+	Converted
	-	-	Converted
	*	*	Converted
	/	/	Converted
	REM	MOD	Modified (1)
EF	SQRT	SQRT_DINT	Modified (2)
	ABS	ABS_DINT	Modified (2)
Operator	INC	INC_DINT	Modified (2)
	DEC	DEC_DINT	Modified (2)
	:=	:=	Converted
Legend:			
(1)	The operator name is modified but the operation remains the same.		
(2)	Replaced by an EF whose name depends on the type of data processed.		

Real words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (Floating point)	**Unity Pro (REAL)**	**Status**
Operator	+	+	Converted
	-	-	Converted
	*	*	Converted
	/	/	Converted

	PL7 (Floating point)	Unity Pro (REAL)	Status
EF	SQRT	SQRT_REAL	Modified (1)
	ABS	ABS_REAL	Modified (1)
	ACOS	ACOS_REAL	Modified (1)
	ASIN	ASIN_REAL	Modified (1)
	ATAN	ATAN_REAL	Modified (1)
	COS	COS_REAL	Modified (1)
	EXP	EXP_REAL	Modified (1)
	EXPT	EXPT_REAL	Modified (1)
	LN	LN_REAL	Modified (1)
	LOG	LOG_REAL	Modified (1)
	SIN	SIN_REAL	Modified (1)
	TAN	TAN_REAL	Modified (1)
Operator	:=	:=	Converted
Legend:			
(1)	The function name is modified but the operation remains the same.		

Logic instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD)	**Unity Pro (INT)**	**Status**
Operator	AND	AND	Converted
	OR	OR	Converted
	XOR	XOR	Converted
	NOT	NOT	Converted

Long integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (DWORD)	**Unity Pro (DINT)**	**Status**
Operator	AND	AND	Converted
	OR	OR	Converted
	XOR	XOR	Converted
	NOT	NOT	Converted

Shift instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD)	**Unity Pro (INT)**	**Status**
EF	SHL	SHL_INT	Modified (1)
	SHR	SHRZ_INT	Modified (1)
	ROL	ROL_INT	Modified (1)
	ROR	ROR_INT	Modified (1)
	ROLW	ROL_INT	Modified (2)
	RORW	ROR_INT	Modified (2)
	WSHL_RBIT	SHL_RBIT_INT	Modified (2)
	WSHR_RBIT	SHR_RBIT_INT	Modified (2)
	WSHRZ_C	SHRZ_RBIT_INT	Modified (2)
Legend:			
(1)	Replaced by an EF whose name depends on the type of operand.		
(2)	The function name is modified but the operation remains the same.		

Long integer words

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD)	**Unity Pro (INT)**	**Status**
EF	SHL	SHL_DINT	Modified (1)
	SHR	SHRZ_DINT	Modified (1)
	ROL	ROL_DINT	Modified (1)
	ROR	ROR_DINT	Modified (1)
	SHL_DWORD	SHL_DWORD	Converted
	SHR_DWORD	SHR_DWORD	Converted
	ROL_DWORD	ROL_DWORD	Converted
	ROR_DWORD	ROR_DWORD	Converted
	DSHL_RBIT	SHL_RBIT_DINT	Modified (2)
	DSHR_RBIT	SHR_RBIT_DINT	Modified (2)
	DSHRZ_C	SHRZ_RBIT_DINT	Modified (2)
	ROLD	ROL_DINT	Modified (2)
	RORD	ROR_DINT	Modified (2)
Legend:			
(1)	Replaced by an EF whose name depends on the type of operand.		
(2)	The function name is modified but the operation remains the same.		

Numerical conversion instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	BCD_TO_INT	BCD_TO_INT	Converted
	INT_TO_BCD	INT_TO_BCD	Converted
	GRAY_TO_INT	GRAY_TO_INT	Converted
	INT_TO_REAL	INT_TO_REAL	Converted
	DINT_TO_REAL	DINT_TO_REAL	Converted
	DINT_TO_DWORD	DINT_TO_DWORD	Converted
	DINT_TO_INT	DINT_TO_INT	Converted
	DINT_TO_WORD	DINT_TO_WORD	Converted
	DWORD_TO_DINT	DWORD_TO_DINT	Converted
	DWORD_TO_INT	DWORD_TO_INT	Converted
	DWORD_TO_WORD	DWORD_TO_WORD	Converted
	INT_TO_DINT	INT_TO_DINT	Converted
	INT_TO_WORD	INT_TO_WORD	Converted
	REAL_TO_DINT	REAL_TO_DINT	Converted
	REAL_TO_INT	REAL_TO_INT	Converted
	WORD_TO_DINT	WORD_TO_DINT	Converted
	WORD_TO_INT	WORD_TO_INT	Converted
	WORD_TO_DWORD	WORD_TO_DWORD	Converted
	TRUNC	REAL_TRUNC_INT REAL_TRUNC_DINT	Modified (1)
	DEG_TO_RAD	DEG_TO_RAD	Converted

	PL7	Unity Pro	Status
EF	RAG_TO_DEG	RAG_TO_DEG	Converted
	CONCATW	INT_AS_DINT	Modified (2)
	DBCD_TO_DINT	DBCD_TO_DINT	Converted
	DBCD_TO_INT	DBCD_TO_INT	Converted
	DINT_TO_DBCD	DINT_TO_DBCD	Converted
	HW	HIGH_INT	Modified (2)
	INT_TO_DBCD	INT_TO_DBCD	Converted
	LW	LOW_INT	Modified (2)
Legend:			
(1)	Replaced by an EF whose name depends on the type of operand.		
(2)	The function name is modified but the operation remains the same.		

Table instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Integer and long integer word tables: instructions

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD or DWORD)	**Unity Pro (INT or DINT)**	**Status**
Operator between two tables	:=	:=	Converted
Operator between a table and an integer word or a long integer word	:=	MOVE_INT_ARINT MOVE_DINT_ARDINT	Modified (1)
Operator between two tables	+	EF: ADD_ARINT ADD_ARDINT	Modified (2)
	-	EF: SUB_ARINT SUB_ARDINT	Modified (2)
	*	EF: MUL_ARINT MUL_ARDINT	Modified (2)
	/	EF: DIV_ARINT DIV_ARDINT	Modified (2)
Operator between two tables	REM	EF: MOD_ARINT MOD_ARDINT	Modified (2)
Operator between a table and an integer word or a long integer word	+	EF: ADD_ARINT_INT ADD_ARDINT_DINT	Modified (2)
Operator between a table and an integer word or a long integer word	-	EF: SUB_INT_ARINT SUB_ARINT_INT SUB_DINT_ARDINT SUB_ARDINT_DINT	Modified (1)
Operator between a table and an integer word or a long integer word	*	EF: MUL_ARINT_INT MUL_ARDINT_DINT	Modified (2)

	PL7 (WORD or DWORD)	**Unity Pro (INT or DINT)**	**Status**
Operator between a table and an integer word or a long integer word	/	EF: DIV_INT_ARINT DIV_ARINT_INT DIV_DINT_ARDINT DIV_ARDINT_DINT	Modified (1)
Operator between a table and an integer word or a long integer word	REM	EF: MOD_INT_ARINT MOD_ARINT_INT MOD_DINT_ARDINT MOD_ARDINT_DINT	Modified (1)
EF	SUM	SUM_ARINT or SUM_ARDINT	Modified (3)
EF	EQUAL	EQUAL_ARINT or EQUAL_ARDINT	Modified (5)

	PL7 (WORD or DWORD)	Unity Pro (INT or DINT)	Status
EF	FIND_EQD	FIND_EQ_ARDINT	Modified (4)
	FIND_EQDP	FIND_EQP_ARDINT	Modified (5)
	FIND_EQW	FIND_EQ_ARINT	Modified (4)
	FIND_EQWP	FIND_EQP_ARINT	Modified (5)
	FIND_GTD	FIND_GT_ARDINT	Modified (4)
	FIND_GTW	FIND_GT_ARINT	Modified (4)
	FIND_LTD	FIND_LT_ARDINT	Modified (4)
	FIND_LTW	FIND_LT_ARINT	Modified (4)
	LENGTH_ARD	LENGTH_ARDINT	Modified (4)
	LENGTH_ARW	LENGTH_ARINT	Modified (4)
	MAX_ARD	MAX_ARDINT	Modified (4)
	MAX_ARW	MAX_ARINT	Modified (4)
	MIN_ARD	MIN_ARDINT	Modified (4)
	MIN_ARW	MIN_ARINT	Modified (4)
	OCCUR_ARD	OCCUR_ARDINT	Modified (4)
	OCCUR_ARW	OCCUR_ARINT	Modified (4)
	ROL_ARD	ROL_ARDINT	Modified (4)
	ROL_ARW	ROL_ARINT	Modified (4)
	ROR_ARD	ROR_ARDINT	Modified (4)
	ROR_ARW	ROR_ARINT	Modified (4)
	SORT_ARD	SORT_ARDINT	Modified (4)
	SORT_ARW	SORT_ARINT	Modified (4)
Legend:			
(1)	The operator is replaced by an EF, which depends on the type of operand.		
(2)	The operator is replaced by an EF, which depends on the type of operand *(see page 177)*.		
(3)	Replaced by an EF whose name depends on the type of operand.		
(4)	The function name is modified but the operation remains the same.		
(5)	The function name is modified, but the operation remains the same except in the case of negative ranks *(see page 178)*.		

Integer and long integer word tables: logic instructions

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7 (WORD or DWORD)	**Unity Pro (INT or DINT)**	**Status**
Operator	NOT	EF: NOT_ARINT NOT_ARDINT	Modified (1)
Operator between two tables	AND	EF: AND_ARINT AND_ARDINT	Modified (1)
	OR	EF: OR_ARINT OR_ARDINT	Modified (1)
	XOR	EF: XOR_ARINT XOR_ARDINT	Modified (1)
Operator between a table and an integer word or a long integer word	AND	EF: AND_ARINT_INT AND_ARDINT_DINT	Modified (1)
Operator between a table and an integer word or a long integer word	OR	EF: OR_ARINT_INT OR_ARDINT_DINT	Modified (1)
Operator between a table and an integer word or a long integer word	XOR	EF: XOR_ARINT_INT XOR_ARDINT_DINT	Modified (1)
Legend:			
(1)	The operator is replaced by an EF, which depends on the type of operand *(see page 177)*.		

Bit tables: logic instructions

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	AND_ARX	AND_AREBOOL	Modified (1)
	NOT_ARX	NOT_AREBOOL	Modified (1)
	OR_ARX	OR_AREBOOL	Modified (1)
	XOR_ARX	XOR_AREBOOL	Modified (1)
Legend:			
(1)	The function name is modified but the operation remains the same.		

Floating point tables: instructions

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
Operator between two tables	:=	:=	Converted
Operator between a table and a real word	:=	MOVE_REAL_ARREAL	Modified (1)
EF	EQUAL_ARR	EQUAL_ARREAL	Modified (3)
	FIND_EQR	FIND_EQ_ARREAL	Modified (2)
	FIND_EQRP	FIND_EQP_ARREAL	Modified (3)
	FIND_GTR	FIND_GT_ARREAL	Modified (2)
	FIND_LTR	FIND_LT_ARREAL	Modified (2)
	LENGTH_ARR	LENGTH_ARREAL	Modified (2)
	MAX_ARR	MAX_ARREAL	Modified (2)
	MIN_ARR	MIN_ARREAL	Modified (2)
	OCCUR_ARR	OCCUR_ARREAL	Modified (2)
	ROL_ARR	ROL_ARREAL	Modified (2)
	ROR_ARR	ROR_ARREAL	Modified (2)
	SORT_ARR	SORT_ARREAL	Modified (2)
	SUM_ARR	SUM_ARREAL	Modified (2)
Legend:			
(1)	The operator is replaced by an EF.		

	PL7	Unity Pro	Status
(2)	The function name is modified but the operation remains the same.		
(3)	The function name is modified, but the operation remains the same except in the case of negative ranks *(see page 178)*.		

Character String Instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
EF	STRING_TO_INT	STRING_TO_INT	Converted
	STRING_TO_DINT	STRING_TO_DINT	Converted
	INT_TO_STRING	INT_TO_STRING	Converted
	DINT_TO_STRING	DINT_TO_STRING	Modified (3)
	STRING_TO_REAL	STRING_TO_REAL	Converted
	REAL_TO_STRING	REAL_TO_STRING	Converted
Operators	>, <, ≤, ≥, =, ≠	>, <, ≤, ≥, =, ≠	Converted
EF	FIND	FIND_INT	Modified (1)
	EQUAL_STR	EQUAL_STR	Converted
	LEN	LEN_INT	Modified (1)
	MID	MID_INT	Modified (2)
	INSERT	INSERT_INT	Modified (2)
	DELETE	DELETE_INT	Modified (2)
	CONCAT	CONCAT_STR	Modified (1)
	REPLACE	REPLACE_INT	Modified (2)
	LEFT	LEFT_INT	Modified (2)
	RIGHT	RIGHT_INT	Modified (2)
	ROUND	STR_ROUND	Modified (1)

Legend:	
(1)	The function name is modified but the operation remains the same.
(2)	In order to comply with the IEC standard, the function name is modified and the operation stays the same except in extreme cases. Refer to the documentation on these EF for further details.
(3)	The function name is not modified and the operation stays the same except in extreme cases. Refer to the documentation on this EF for further details. Converting a variable into a character string is displayed differently when using PL7 software or Unity Pro: ● With Unity Pro, the character string is 11 characters long. ● With PL7, the character string is 12 characters long.

Time management instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	RRTC	RRTC	Converted
	WRTC	WRTC	Converted
	PTC	PTC	Converted
	ADD_TOD	ADD_TOD_PL7	Modified (1)
	ADD_DT	ADD_DT_PL7	Modified (1)
	DELTA_TOD	DELTA_TOD	Converted
	DELTA_D	DELTA_D	Converted
	DELTA_DT	DELTA_DT	Converted
	SUB_TOD	SUB_TOD_PL7	Modified (1)
	SUB_DT	SUB_DT_PL7	Modified (1)
	DAY_OF_WEEK	DAY_OF_WEEK	Converted
	TRANS_TIME	TRANS_TIME	Converted
	DATE_TO_STRING	DATE_DINT_TO_STRING	Modified (1)
	TOD_TO_STRING	TOD_DINT_TO_STRING	Modified (1)
	DT_TO_STRING	DT_ARINT_TO_STRING	Modified (1)
	TIME_TO_STRING	TIME_DINT_TO_STRING	Modified (1)
	SCHEDULE	SCHEDULE	Converted
Legend:			
(1)	The function name is modified but the operation remains the same.		

Exchange Instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction List

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
EF	READ_STS	READ_STS	Modified (1)
	WRITE_CMD	WRITE_CMD	Modified (1)
	READ_PARAM	READ_PARAM	Modified (1)
	WRITE_PARAM	WRITE_PARAM	Modified (1)
	RESTORE_PARAM	RESTORE_PARAM	Modified (1)
	SAVE_PARAM	SAVE_PARAM	Modified (1)
	SMOVE	SMOVE	Modified (1)
	XMOVE	XMOVE	Modified (1)
	LXM_SAVE	LXM_SAVE	Modified (1)(2)
	LXM_RESTORE	LXM_RESTORE	Modified (1)
Key:			
(1)	The function is replaced by an EF. The parameters are entered between brackets (example: `READ_STS(%CH0.4.0)`).		
(2)	The order of the function parameters has been modified *(see page 179)*.		

Input/output instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
EF	MOD_CAM	MOD_CAM	Modified (1)
	MOD_PARAM	MOD_PARAM	Modified (1)
	MOD_TRACK	MOD_TRACK	Modified (1)
	TRF_RECIPE	TRF_RECIPE	Modified (1)
	DETAIL_OBJECT	DETAIL_OBJECT	Modified (1)
Legend:			
(1)	The function is replaced by an EF. The parameters are entered between brackets (example: `MOD_CAM(%CH0.2.0,1,2,9)`).		

Process control instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
EF	PID	PID_INT	Modified (1)
	PID_MMI	-	Deleted
	PWM	PWM_INT	Modified (1)
	SERVO	SERVO_INT	Modified (1)
Legend:			
(1)	The function name is modified: its operation remains the same but the order of its parameters has been changed *(see page 179)*.		

Other instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	FTOF	FTOF	Modified (1)
	FTON	FTON	Modified (1)
	FTP	FTP	Modified (1)
	SCOUNT	SCOUNT	Modified (1)
	MASKEVT	MASKEVT	Converted (2)
	UNMASKEVT	UNMASKEVT	Converted (2)
	FPULSOR	FPULSOR	Modified (1)
	READ_PCMCIA	READ_PCMCIA	Modified (3)
	READ_PCMEXT	READ_PCMCIA	Modified (4)
	SET_PCMCIA	SET_PCMCIA	Modified (3)
	SET_PCMEXT	SET_PCMCIA	Modified (4)
	WRITE_PCMCIA	WRITE_PCMCIA	Modified (3)
	WRITE_PCMEXT	WRITE_PCMCIA	Modified (4)
Legend:			
(1)	The order of the function parameters has been modified *(see page 179)*.		
(2)	In Unity Pro, the operation and comparison blocks of Instruction List language written in Structured Text language *(see page 155)* are not converted. Consequently, the MASKEVT and UNMASKEVT instructions located in one of these blocks are not converted.		
(3)	The function name remains the same but a parameter is added in the first position. When the call is made the value 0 must be entered in this parameter.		
(4)	The function name is modified but the operation remains the same.		

NOTE: `FTOF`, `FTON` and `FTP` are not converted for M340 targets.

Communication instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	CALL_MODEM	CALL_MODEM	Converted
	CANCEL	CANCEL	Converted
	DATA_EXCH	DATA_EXCH	Modified (1)
	INPUT_CHAR	INPUT_CHAR	Modified (1)
	OUT_IN_CHAR	OUT_IN_CHAR	Modified (1)
	PRINT_CHAR	PRINT_CHAR	Modified (1)
	RCV_TLG	RCV_TLG	Modified (1)
	READ_ASYN	READ_ASYN	Modified (1)
	READ_GDATA	READ_GDATA	Modified (1)
	READ_VAR	READ_VAR	Modified (1)
	ROR1_ARB	ROR1_ARB	Converted
	SEND_REQ	SEND_REQ	Modified (1)
	SEND_TLG	SEND_TLG	Converted
	SERVER	UNITE_SERVER	Modified (1) (2)
	SWAP	SWAP_ARINT	Modified (2)
	WRITE_ASYN	WRITE_ASYN	Converted
	WRITE_GDATA	WRITE_GDATA	Converted
	WRITE_VAR	WRITE_VAR	Converted
Legend:			
(1)	The order of the function parameters has been modified *(see page 179)*.		
(2)	The function name is modified but the operation remains the same.		

TCP Open instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list: TCPIP_LEVEL1

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	FCT_ACCEPT	-	Deleted
	FCT_BIND	-	Deleted
	FCT_LISTEN	-	Deleted
	FCT_RECEIVE	-	Deleted
	FCT_SELECT	-	Deleted
	FCT_CONNECT	FCT_CONNECT	Converted
	FCT_SEND	-	Deleted
	FCT_SHUTDOWN	-	Deleted
	FCT_SOCKET	-	Deleted
	FCT_SETSOCKOPT	-	Deleted

Instruction list: TCPIP_DFB

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	FCT_ACCEPT_DFB	FCT_ACCEPT	Modified (1)
	FCT_BIND_DFB	FCT_BIND	Modified (1)
	FCT_LISTEN_DFB	FCT_LISTEN	Modified (1)
	FCT_RECEIVE_DFB	FCT_RECEIVE	Modified (1)
	FCT_SELECT_DFB	FCT_SELECT	Modified (1)
	FCT_SEND_DFB	FCT_SEND	Modified (1)
	FCT_SHUTDOWN_DFB	FCT_SHUTDOWN	Modified (1)
	FCT_SOCKET_DFB	FCT_SOCKET	Modified (1)
	FCT_SETSOCKOPT_DFB	FCT_SETSOCKOPT	Modified (1)
Legend:			
(1)	The function name is modified but the operation remains the same.		

Diagnostics instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	**Unity Pro**	**Status**
EF	DEREG	DEREG	Converted
	REGDFB	REGDFB	Modified (1)
	UREGDFB	UREGDFB	Modified (1)
	REGIO	-	Deleted
Legend:			
(1)	The function interface is modified. You must modify this function manually *(see page 38)* before launching the conversion procedure.		

Grafcet instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	RESET_XIT	-	Deleted

Human Machine Interface (HMI) instructions

Introduction

Conversion replaces the PL7 instructions by their Unity Pro equivalent.

Instruction list

The following table describes any correspondence and differences between PL7 and Unity Pro instructions.

	PL7	Unity Pro	Status
EF	ADJUST	No equivalent	Deleted
	ASK_MSG		
	ASK_VALUE		
	ASSIGN_KEYS		
	CONTROL_LEDS		
	DISPLAY_ALRM		
	DISPLAY_GRP		
	DISPLAY_MSG		
	GET_MSG		
	GET_VALUE		
	PANEL_CMD		
	SEND_ALARM		
	SEND_MSG		

7.4 Correspondences between PL7 and Unity Pro: SFBs

Aim of this sub-section

In Unity Pro, SFBs are replaced by equivalent

- EFBs for Premium applications
- DFBs for Modicon M340 applications. The names of the DFBs are derived from the SFB names.

SFB instances are also converted automatically.

Example: the SFB `%Ti` is translated by an EFB of instance name `Ti` and type `PL7_3_TIMER`.

This sub-section therefore contains the tables showing correspondence between SFBs and EFBs.

NOTE: Where a PL7 SFB occurs in Unity Pro in an ST section of a Modicon M340 application, a conversion error message is generated *(see page 46)*.

What's in this Section?

This section contains the following topics:

Topic	Page
Types of Unity Pro Function Block Instances of SFBs	128
Behavior of Converted SFBs in Modicon M340 Applications	129
Call of SFB Substitutions in Structured Text	130
Call of an SFB in instruction list language	132
Call of an SFB in ladder language	134

Types of Unity Pro Function Block Instances of SFBs

Introduction

For Premium, an SFB instance is converted to an EFB instance.

For Modicon M340, an SFB instance is converted to a DFB instances.

The DFB types used for Modicon M340 have the same name as the EFB types used for Premium.

Types of instances

The following table, which is valid for both families, shows the types of instances.

SFB		**Types of Unity Pro instances**
%T		PL7_3_TIMER
%TM	TON	PL7_TON
	TOF	PL7_TOF
	TP	PL7_TP
%C		PL7_COUNTER
%MN		PL7_MONOSTABLE
%R	%Ri.LEN ≤32	PL7_REGISTER_32
	32 < %Ri.LEN ≤255	PL7_REGISTER_255
%DR		PL7_DRUM (1)
Key:		
(1)	The PL7_DRUM EFB contains an extra output parameter *(see page 181)* compared with the PL7 %DR instance type. **Note:** A PL7_DRUM is converted in Modicon M340 applications to an empty DFB.	

Behavior of Converted SFBs in Modicon M340 Applications

Behavior Must be Checked

Following the conversion from a PL7 project in ST language to a Modicon M340 application, the behavior of converted SFBs must be checked.

Notification about Differences

The conversion of ST language including SFBs is done with a conversion error message to log all the calls of SFBs.

This is done to inform the programmer about differences that may happen, depending on the program itself.

PL7 versus Unity Pro

SFBs in PL7 are managed by the PLC operating system asynchronously with the program.

On the opposite, the converted SFBs are function blocks in Unity Pro, that are managed synchronously with the execution of the program.

Consequences

As a consequence:

- It is mandatory to execute the function block in Unity Pro at each scan time to refresh the current values (elapsed time, output). This is not mandatory in PL7. In case of conditional calls of SFBs, take care to keep a call active as long as the SFB is running or add an unconditional call.
- Check the correct behavior of the SFBs in all operating modes of the PLC (cold start, warm start, PLC initialization, PLC running).

PL7_DRUM

NOTE: A `PL7_DRUM` is converted in Modicon M340 applications to an empty DFB.

Call of SFB Substitutions in Structured Text

Introduction

The call of SFB substitutions in structured text (ST) is different between Premium and Modicon M340.

Call of PL7 SFB substitutions in Unity Pro ST

The following table describes the conversion of SFB substitution calls for Premium and Modicon M340.

SFB	Specific	PL7	Unity Pro - Premium	Unity Pro - Modicon M340
%T		START %Ti	START_PL7_3_TIMER(Ti);	Ti(E:= TRUE),C:= TRUE); (*START*);
		STOP %Ti	STOP_PL7_3_TIMER(Ti);	Ti(E:= TRUE),C:= FALSE); (* STOP*);
		PRESET %Ti	PRESET_PL7_3_TIMER(Ti);	Ti(E:= FALSE),C:= FALSE); (* PRESET*);
%TM	TON	START %TMi	START_PL7_TON(TMi);	TMi(S:=TRUE); (* START*);
		DOWN %TMi	DOWN_PL7_TON(TMi);	TMi(S:=FALSE); (* DOWN*);
	TOF	START %TMi	START_PL7_TOF(TMi);	TMi(S:=TRUE); (* START*);
		DOWN %TMi	DOWN_PL7_TOF(TMi);	TMi(S:=FALSE); (* DOWN*);
	TP	START %TMi	START_PL7_TP(TMi);	TMi(S:=TRUE); (* START*);
		DOWN %TMi	DOWN_PL7_TP(TMi);	TMi(S:=FALSE); (* DOWN*);
%C		RESET %Ci	RESET_PL7_COUNTER(Ci);	Ci(R:=TRUE);Ci(R:=FALSE); (* RESET*);
		PRESET %Ci	PRESET_PL7_COUNTER(Ci);	Ci(P:=TRUE);Ci(P:=FALSE); (* PRESET*);
		UP %Ci	UP_PL7_COUNTER(Ci);	Ci(CU:= TRUE);Ci(CU:= FALSE); (* UP*);
		DOWN %Ci	DOWN_PL7_COUNTER(Ci);	Ci(CD:= TRUE);Ci(CD:= FALSE); (*DOWN*);
%MN		START %MNi	START_PL7_MONOSTABLE(MNi);	MNi(S:=TRUE); (* START*);

SFB	Specific	PL7	Unity Pro - Premium	Unity Pro - Modicon M340
%R	%Ri.LEN≤32	RESET %Ri	RESET_PL7_REGISTER_32(Ri);	Ri(R:=TRUE);Ri(R:=FALSE); (* RESET*);
		PUT %Ri	PUT_PL7_REGISTER_32(Ri);	Ri(I:=TRUE);Ri(I:=FALSE); (* PUT*);
		GET %Ri	GET_PL7_REGISTER_32(Ri);	Ri(O:=TRUE);Ri(O:=FALSE); (* GET*);
	32 < %Ri.LEN≤255	RESET %Ri	RESET_PL7_REGISTER_255(Ri);	Ri(R:=TRUE);Ri(R:=FALSE); (* RESET*);
		PUT %Ri	PUT_PL7_REGISTER_255(Ri);	Ri(I:=TRUE);Ri(I:=FALSE); (* PUT*);
		GET %Ri	GET_PL7_REGISTER_255(Ri);	Ri(O:=TRUE);Ri(O:=FALSE); (* GET*);
%DR		RESET %DRi	RESET_PL7_DRUM(DRi);	DRi(R:=TRUE); DRi(R:=FALSE); (*RESET*);
		UP %DRi	UP_PL7_DRUM(DRi) WORD_TO_BIT (INT_TO_WORD (DRi.W), list of 16 DRUM output objects);	DRi(U:=TRUE); DRi(U:=FALSE); (* UP*); WORD_TO_BIT(INT_TO_WORD(DRi.W), List of 16 DRUM output objects);

NOTE: The DRUM SFB is represented by an empty DFB for M340 targets.

Call of an SFB in instruction list language

Introduction

The conversion automatically replaces SFBs with equivalent EFBs for Premium and DFBs for Modicon M340, insofar as these exist.

Call of an SFB in IL

The following table describes the correspondence and possible differences between SFBs and EFBs (Premium) or DFBs (Modicon M340).

			PL7	Unity Pro	State
SFB	%T		Not used in instruction list language	-	-
SFB	%TM	TON	IN %TMi	CAL TMi(S:=TEMPBOOL)	Converted
		TOF	IN %TMi	CAL TMi(S:=TEMPBOOL)	
		TP	IN %TMi	CAL TMi(S:=TEMPBOOL)	
SFB	%C		R %Ci S %Ci CU %Ci CD %Ci	CAL Ci(CU:=0,CD:=0,R:=TRUE,P:=FALSE) CAL Ci(CU:=0,CD:=0,R:=0,P:=TRUE) CAL Ci(CU:=TRUE,CD:=FALSE,R:=FALSE,P:=FALSE) CAL Ci(CU:=FALSE,CD:=TRUE,R:=FALSE,P:=FALSE)	Converted
SFB	%MN		S %MNi	CAL MNi(S:=FALSE)	Converted
SFB	%R	%Ri.LEN ≤32	R %Ri I %Ri O %Ri	CAL Ri(R:=TRUE) CAL Ri(R:=0,I:=TRUE,O:=0) CAL Ri(R:=0,I:=0,O:=TRUE)	Converted
		32 < %Ri.LEN ≤ 255	R %Ri I %Ri O %Ri	CAL Ri(R:=TRUE) CAL Ri(R:=0,I:=TRUE,O:=0) CAL Ri(R:=0,I:=0,O:=TRUE)	

		PL7	Unity Pro	State
SFB	%DR	R %DRi U %DRi	CAL DRi(R:=TRUE,U:=0) **(1)** CAL DRi(R:=0,U:=TRUE) **(1)**	Converted (Premium only)
	Note: For Modicon M340, the converter replaces a DRUM SFB with an empty DFB.			
Key:				
(1)	You must also assign the values of the current step to the DRUM output objects *(see page 181)*.			

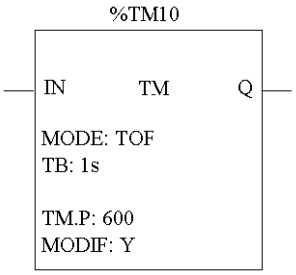
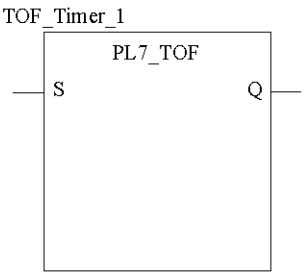
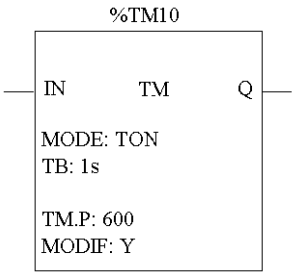
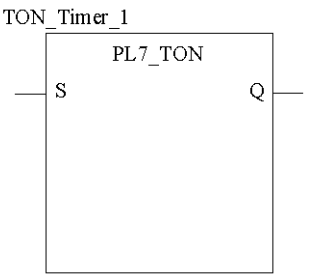
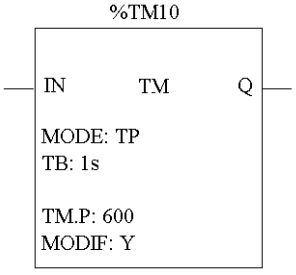
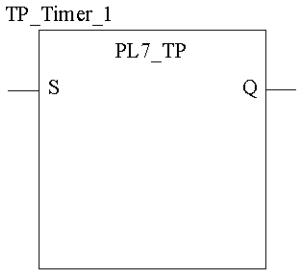
Call of an SFB in ladder language

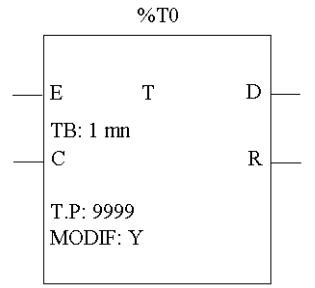
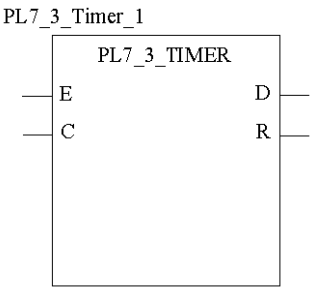
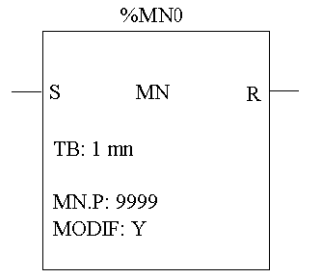
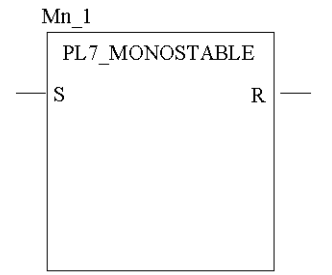
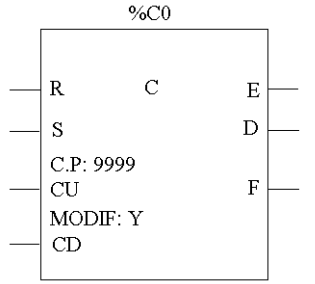
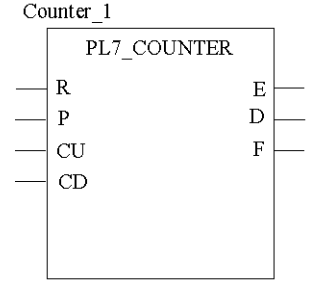
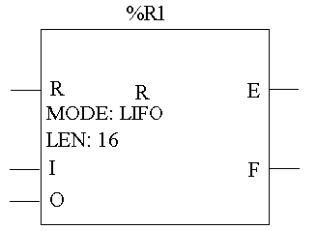
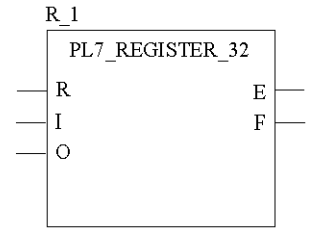
Introduction

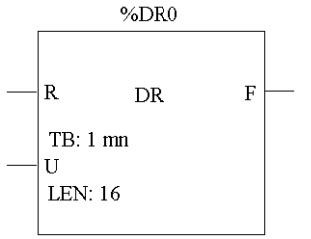
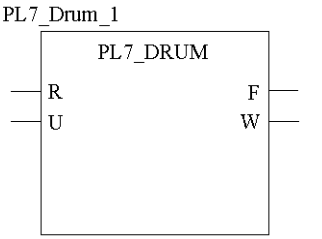
The conversion automatically replaces SFBs with equivalent EFBs (Premium) or DFBs (Modicon M340), insofar as these exist.

Call of an SFB in Ladder

The following table describes the correspondence and possible differences between SFBs and EFBs (Premium) or DFBs (Modicon M340).

		PL7	Unity Pro	State
SFB	PL7 Timer			Modified
				
				

		PL7	Unity Pro	State
SFB	PL7_3 Timer			Modified
SFB	PL7 Monostable			Modified
SFB	PL7 Counter			Modified
SFB	PL7 Register		 (1)	Modified

		PL7	Unity Pro	State
SFB	PL7 Drum		 (2)	Modified (Premium only)
	Note: For Modicon M340, the converter replaces a `DRUM` SFB with an empty DFB.			
Key:				
(1)	Depending on the register length *(see page 128)*, can be converted into `PL7_REGISTER_255`.			
(2)	You must also assign the values of the current step to the DRUM output objects *(see page 181)*.			

Correspondences between ladder language elements

8

Subject of this Chapter

This chapter contains the tables of correspondence between ladder language elements

The PL7 ladder language elements are divided into two categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
The definition of a ladder network	138
The rungs	139
Coils	140
Operate blocks and compare blocks	141
Conversion restrictions: PL7 ladder language	142

The definition of a ladder network

Introduction

Conversion replaces the PL7 ladder language elements by their Unity Pro equivalent.

Definition of a ladder network

The following table describes any correspondence and differences between PL7 and Unity Pro ladder language objects.

	PL7	Unity Pro	Status
Label	%Li	Li	Modified
Comment	Network comment	Network comment	Converted

The rungs

Introduction

Conversion replaces the PL7 ladder language elements by their Unity Pro equivalent.

Rungs

The following table describes any correspondence and differences between PL7 and Unity Pro ladder language graphic objects.

Type of rung	PL7 Graphic representation	Unity Pro Graphic representation	Status
Direct	--I I--	--I I--	Converted
Reverse	--I / I--	--I / I--	Converted
Rising edge	--I P I--	--I P I--	Converted
Falling edge	--I N I--	--I N I--	Converted

Coils

Introduction

The conversion replaces the PL7 ladder language elements with their Unity Pro equivalent.

Coils

The following table describes the correspondence and possible differences between PL7 and Unity Pro graphic objects.

	PL7 Graphical representation	**Unity Pro Graphical representation**	**State**
Direct coil	--( )--	--( )--	Converted
Negated coil	--(/)--	--(/)--	Converted
Set coil	--(S)--	--(S)--	Converted
Reset coil	--(R)--	--(R)--	Converted
Jumps to a label	-->>%Li	-->>Li	Converted
Subroutine return	--<RETURN>--	--<RETURN>--	Converted
Program halt coil	--<HALT>--	EF	Modified (1)
Coil #	--(#)--	--(^{name})--	Modified (2)
SR call coil	--(C)--	EF	Modified (1)
Key:			
(1)	The coil is replaced by an EF.		
(2)	The coil # is replaced by a direct coil with the transition name (^{name}) superimposed.		

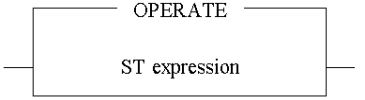
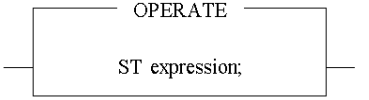
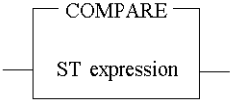
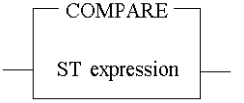
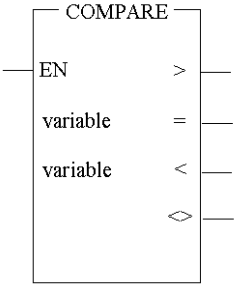
Operate blocks and compare blocks

Introduction

The conversion replaces the PL7 ladder language elements with their Unity Pro equivalent.

Operate blocks and compare blocks

The following table describes the correspondence and possible differences between PL7 and Unity Pro ladder language graphic objects.

	PL7 **Graphical representation**	**Unity Pro** **Graphical representation**	**State**
Operate block			Modified (1)
Horizontal compare block			Converted
Vertical compare block		EF	Modified (2)

Key:	
(1)	A semi-colon is added to the end of the ST expression.
(2)	The operate block is replaced by an EF.

Conversion restrictions: PL7 ladder language

Conversion of blocks

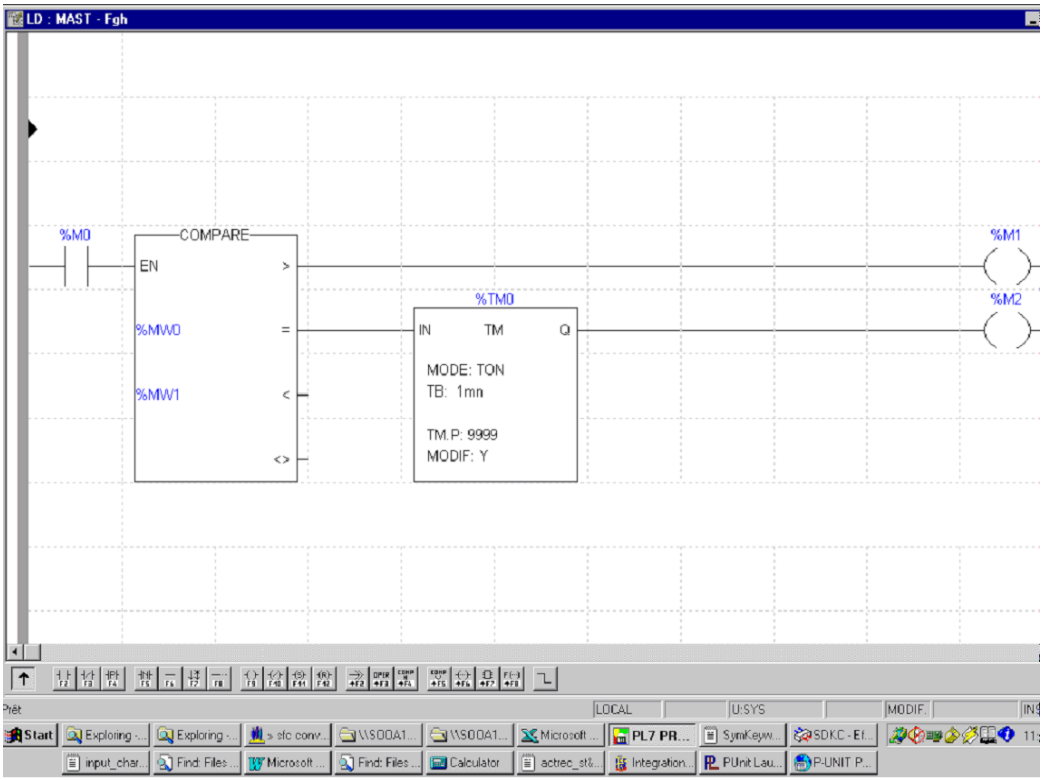
The blocks in Unity Pro ladder language are taller than the blocks in PL7 ladder language.

The PL7 application converter automatically reorganizes the height of the blocks to guarantee that the application works correctly.

However, if a block is embedded between two outputs of another block, the procedure for converting the application:

- converts as far as possible,
- displays the partially converted network with errors

The following screen gives an example of a block embedded between two outputs of another block: this part of the program cannot be automatically converted.



Correspondences between Structured Text language elements

9

Subject of this Chapter

This chapter contains the tables of correspondence between structured text language elements.

The PL7 Structured Text language elements are divided into three categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),
- those that have no Unity Pro equivalent (status: deleted).

NOTE: if a PL7 Structured Text language element has no Unity Pro correspondence, both a warning and aconversion error message are generated *(see page 46)*.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	Correspondences between PL7 and Unity Pro: Structured Text language sequences	144
9.2	Correspondences between PL7 and Unity Pro: Structured Text language instructions	145

9.1 Correspondences between PL7 and Unity Pro: Structured Text language sequences

The sequences

Introduction

Conversion replaces the PL7 Structured Text language elements by their Unity Pro equivalent.

Sequences

The following table describes any correspondence and differences between PL7 and Unity Pro Structured Text language sequences.

	PL7	Unity Pro	Status
Label	`%Li`	`Li`	Modified
Comment	Comment linked to the sequence	Comment linked to the sequence	Converted

9.2 Correspondences between PL7 and Unity Pro: Structured Text language instructions

Command instructions

Introduction

Conversion replaces the PL7 Structured Text language elements by their Unity Pro equivalent.

Instruction list

The following table describes any correspondences and differences between PL7 and Unity Pro Structured Text command instructions.

	PL7	**Unity Pro**	**Status**
Instructions	IF	IF	Converted
	CASE	CASE	Converted
	WHILE	WHILE	Converted
	REPEAT	REPEAT	Converted
	EXIT	EXIT	Converted
	FOR	FOR	Modified (1)
	HALT	HALT()	Modified (2)
	JUMP	JUMP	Converted
	SRi	SRi()	Modified (3)
	RETURN	RETURN	Converted
Legend:			
(1)	Conversion of this instruction generates an error message during the analysis phase *(see page 183)*.		
(2)	The instruction is replaced by an EF.		
(3)	This instruction is replaced by a section call.		

Correspondences between Instruction List language elements

10

Subject of this Chapter

This chapter contains the tables of correspondence between Instruction List language elements.

The PL7 Instructions List language elements are divided into three categories:

- those that remain unchanged and are translated automatically (status: converted),
- those that have a Unity Pro equivalent and are translated automatically (status: modified),
- those that have no Unity Pro equivalent (status: deleted).

NOTE: if a PL7 Instruction List language element has no Unity Pro correspondence, both a warning and aconversion error message are generated *(see page 46)*.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	Correspondences between PL7 and Unity Pro: Instruction List language sequences	148
10.2	Correspondences between PL7 and Unity Pro: Instruction List language instructions	149

10.1 Correspondences between PL7 and Unity Pro: Instruction List language sequences

The sequences

Introduction

Conversion replaces the PL7 Instruction List language elements by their Unity Pro equivalent.

Sequences

The following table describes any correspondence and differences between PL7 and Unity Pro Instruction List language sequences.

	PL7	Unity Pro	Status
Label	`%Li`	`Li`	Modified
Comments	Comments linked to the sequence	Comments linked to the sequence	Converted

10.2 Correspondences between PL7 and Unity Pro: Instruction List language instructions

Subject of this Section

This section contains the tables of correspondence between PL7 Instruction List language instructions and their Unity Pro equivalent.

What's in this Section?

This section contains the following topics:

Topic	Page
Command instructions	150
Boolean instructions	151
Instruction List language extensions	152

Command instructions

Introduction

Conversion replaces the PL7 Instruction List language elements by their Unity Pro equivalent.

Instruction list

The following table describes any correspondences and differences between PL7 and Unity Pro Instruction List command instructions.

	PL7	**Unity Pro**	**Status**
Instructions	HALT	HALT	Modified (1)
	HALTC	ST ACCU HALT (EN:=ACCU)	Modified
	HALTCN	STN ACCU HALT (EN:=ACCU)	Modified
	END	END	Modified (1)
	ENDC	ST ACCU END (EN:=ACCU)	Modified
	ENDCN	STN ACCU END (EN:=ACCU)	Modified
	JMP	JMP	Converted
	JMPC	JMPC	Converted
	JMPCN	JMPCN	Converted
	RET	RET	Converted
	RETC	RETC	Converted
	RETCN	RETCN	Converted
	SRi	CAL SRi	Modified (2)
	NOP	-	Deleted
Edge management (3)	F	FE	Modified
	R	RE	Modified
Legend:			
(1)	The instruction is replaced by an EF.		
(2)	This instruction is replaced by a conditional section call.		
(3)	In Unity Pro, edges are managed by EFs.		

Boolean instructions

Introduction

Conversion replaces the PL7 Instruction List language elements by their Unity Pro equivalent.

Instruction list

The following table describes any correspondences and differences between the Boolean instructions of PL7 and Unity Pro Instruction List language.

	PL7	**Unity Pro**	**Status**
Instructions	LD	LD	Converted
	LDF	LDF	Converted
	LDN	LDN	Converted
	LDR	LDR	Converted
	ST	ST	Converted
	STN	STN	Converted
	AND	AND	Converted
	OR	OR	Converted
	XOR	XOR	Converted
	)	)	Converted
	N	N	Converted
	R	RESET	Modified (1)
	S	SET	Modified (1)
	MPS, MRD, MPP	-	Modified (2)
Legend:			
(1)	These instructions are replaced by an EF. *(see Unity Pro, Standard, Block Library)*		
(2)	The instructions MPS, MRD and MPP do not exist in Unity Pro *(see page 185)*.		

Instruction List language extensions

Introduction

PL7 IL language has the extension to support ST statements in IL, an equivalent of the operation and comparison blocks in LD language.

The IL language in Unity Pro does not support these extension.

So the ST statements are converted into the corresponding IL language.

This service is applicable for all PL7 applications for TSX Premium and TSX Micro.

The next pages give some examples of the conversion.

To help the user to understand the conversion, a comment is added to the program, corresponding to the instruction in PL7.

Example 1

Variables of the following example in PL7 and in Unity Pro

-	PL7		Unity Pro)	
Variable Name	**Address**	**Type**	**Address**	**Type**
Var_word_201	%KW0	WORD	%KW0	INT
Var_word_202	%KW1	WORD	%KW1	INT
Var_word_203	%KW3	WORD	%KW3	INT
Var_word_204	%KW5	WORD	%KW5	INT
Var_word_205	%KW7	WORD	%KW7	INT
Var_bool_001	%MW421:X2	BOOL	%MW421.2	BOOL
Var_word_206	%MW50	WORD	%MW50	INT
Var_word_207	%MW51	WORD	%MW51	INT
Var_ebool_101	%M122	EBOOL	%M122	EBOOL
Var_word_208	%MW305	WORD	%MW305	INT
Var_word_209	%MW306	WORD	%MW306	INT

Operation and comparison block conversion to Unity Pro

PL7	Unity Pro
LD %MW421:X2	LD Var_bool_001
AND [%MW50>0]	(*[[%MW50>0]]*) AND (Var_word_206 GT 0)
AND [%MW50<6]	(*[[%MW50<6]]*) AND (Var_word_206 LT 6)
AND [%MW51>0]	(*[[%MW51>0]]*) AND (Var_word_207 GT 0)
AND [%MW51<9]	(*[[%MW51<9]]*) AND (Var_word_207 LT 9)
ST %M122	ST Var_ebool_101
LD %M122	LD Var_ebool_101
[%MW305:=(%MW50-%KW7)*%KW1+%KW5]	JMPCN LAB1 (*[%MW305:=(%MW50-%KW7)*%KW1+%KW5]*) LD Var_word_206 SUB Var_word_205 MUL Var_word_202 ADD Var_word_204 ST Var_word_208
[%MW306:=(%MW51-%KW7)*%KW1+%KW3]	(*[%MW306:=(%MW51-%KW7)*%KW1+%KW3]*) LD Var_word_207 SUB Var_word_205 MUL Var_word_202 ADD Var_word_203 ST Var_word_209
[%MW0[%MW306]:63:=%KW0[%MW305]:63]	(*[%MW0[%MW306]:63:=%KW0[%MW305]:63]*) LD %KW0[Var_word_208]:63 ST %MW0[Var_word_209]:63 LAB1:

Example 2

Conversion of SFBs to IL language

PL7	Unity Pro
PROGRAM (*PHRASE*) (* block timer *) BLK %TM0 LD %M0 IN OUT_BLK LD Q ST %M1 END_BLK (*END_PHRASE*)	(* block timer *) (*BLK Tm0*) LD M0 ST Tm0.S CAL Tm0 (*OUT_BLK*) LD Tm0.Q ST M1
(*PHRASE*) (* block monostable *) BLK %MN0 LD %M0 S OUT_BLK LD R ST %M2 END_BLK (*END_PHRASE*)	(* block monostable *) (*BLK Mn0*) LD M0 ST Mn0.S CAL Mn0 (*OUT_BLK*) LD Mn0.R ST M2
(*PHRASE*) (* block register *) BLK %R0 LD %M0 R LD %M9 I LD %M10 O OUT_BLK LD E ST %M11 LD F ST %M12 END_BLK (*END_PHRASE*)	(* block register *) (*BLK R0*) LD M0 ST R0.R LD M9 ST R0.I LD M10 ST R0.O CAL R0 (*OUT_BLK*) LD R0.E ST M11 LD R0.F ST M12

PL7	Unity Pro
(*PHRASE*) (* block drum *) BLK %DR0 LD %M0 R LD %M13 U OUT_BLK LD F ST %M13 END_BLK (*END_PHRASE*) END_PROGRAM	(* block drum *) (*BLKDr0*) LD M0 ST Dr0.R LD M13 ST Dr0.U CAL Dr0 (*OUT_BLK*) LD Dr0.F ST M13
(*PHRASE*) (* block counter *) BLK %C0 LD %M0 R LD %M6 S LD %M7 CU LD %M8 CD OUT_BLK LD E ST %M3 LD D ST %M4 LD F ST %M5 END_BLK (*END_PHRASE*)	(* block counter *) (*BLK C0*) LD M0 ST C0.R LD M6 ST C0.P LD M7 ST C0.CU LD M8 ST C0.CD CAL C0 (*OUT_BLK*) LD C0.E ST M3 LD C0.D ST M4 LD C0.F ST M5

IL extensions

The following table describes the Instruction List language extensions which have no Unity Pro equivalents.

	PL7	Unity Pro	Status
	[OF]	-	Deleted (1)
	MASKEVT		
	UNMASKEVT		
	[...expression...]		
Legend:			
(1)	It is necessary to manually replace this part of the application.		

Correspondences between Grafcet language elements

11

Aim of this section

This section contains the tables showing correspondence between Grafcet language elements.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Grafcet instructions	158
Conversion Settings for PL7 Grafcet Elements	159
Conversion restrictions: PL7 Grafcet language	160

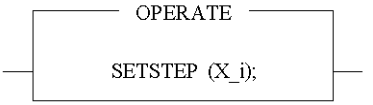
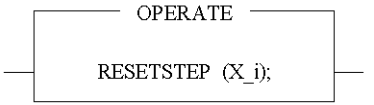
Grafcet instructions

Introduction

The conversion replaces the PL7 Grafcet language elements with their Unity Pro equivalent.

Grafcet instructions in ladder language

The following table describes the correspondence and possible differences between PL7 and Unity Pro ladder language Grafcet instructions.

	PL7	Unity Pro	State
Instructions			Modified
			Modified

Grafcet instructions in structured text

The following table describes the correspondence and possible differences between PL7 and Unity Pro structured text Grafcet instructions.

	PL7	Unity Pro	State
Instructions	`SET %Xi`	`SETSTEP (X_i);`	Modified
	`RESET %Xi`	`RESETSTEP (X_i);`	Modified

Grafcet instructions in instruction list language

The following table describes the correspondence and possible differences between PL7 and Unity Pro instruction list language Grafcet instructions.

	PL7	Unity Pro	State
Instructions	`S %Xi`	`CAL SETSTEP (X_i)`	Modified
	`R %Xi`	`CAL RESETSTEP (X_i)`	Modified

Conversion Settings for PL7 Grafcet Elements

Overview

The **Conversion Settings** tab, which is accessed in Unity Pro via **Tools** →**Options** contains options for the conversion of PL7 Grafcet elements.

The options that can be enabled/disabled by checkboxes are listed below.

Transition Optimization

- When this checkbox is enabled, in order to improve the performances at execution time, some SFC transitions are converted to variables.
- When this checkbox is **not** enabled, the transitions are converted as they were in PL7.

Step State Explicit Definition

- If this checkbox is enabled, when the conversion of PL7 application to Unity Pro is done, the macro steps not used on the PL7 program are declared on the Unity Pro project.
- If this checkbox is **not** enabled the macro steps not used on the PL7 program are not converted to the Unity Pro project.

Symbol As Step Name

- When this checkbox is enabled, the PL7 steps %Xi are converted with the PL7 step symbol. For example on PL7 %X2 with the symbol Motor_1 is converted into Motor_1.
- When this checkbox is **not** enabled, the PL7 steps %Xi are converted on X_i. For example on PL7 %X2 is converted into X_2.

Conversion restrictions: PL7 Grafcet language

Size of programming in SFC language

Diagrams in PL7 Grafcet language are automatically restructured by the PL7 application converter into diagrams in Unity Pro SFC language.

The corresponding Unity Pro SFC language has a single programming page with a maximum of 200 lines.

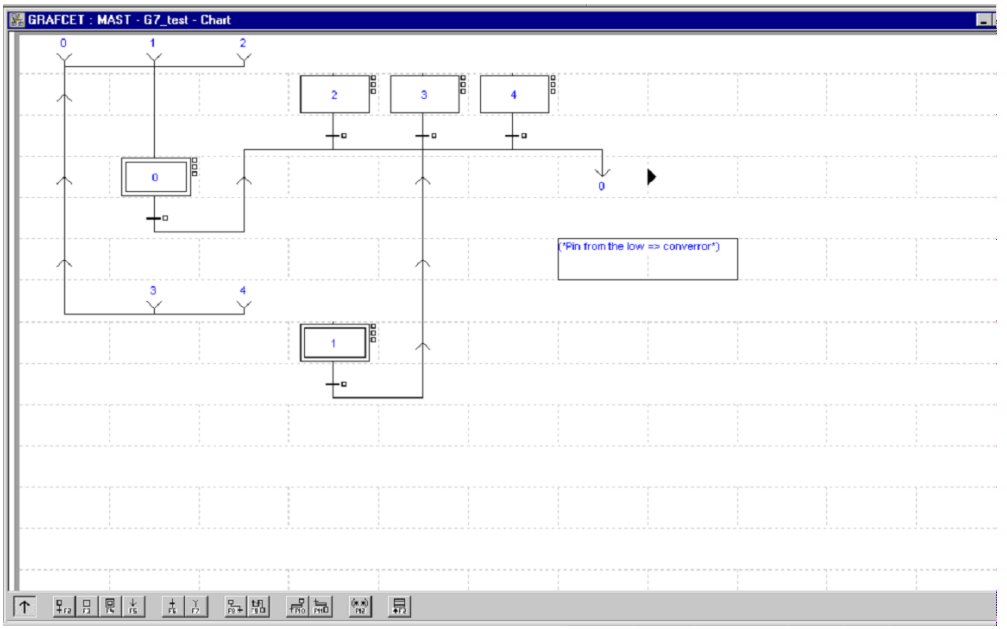
In the rare cases when the Grafcet programming exceeds this limits, a Converror message *(see page 49)* will be displayed in the output window.

Link from bottom to top

In Unity Pro SFC language, it is not possible to have several links going towards the same connector. The PL7 application converter automatically replaces this configuration with an equivalent Unity Pro configuration.

However, it is impossible, in SFC language, to have a **link from bottom to top**. If your PL7 application to be converted contains this type of link, you must manually complete the diagram using the Converror message *(see page 49)* that is displayed in the output window.

The following screen gives an example of a link from bottom to top drawn up in PL7 Grafcet language: this part of the program cannot be automatically converted.



Other correspondences between PL7 and Unity Pro elements

12

Printouts, animation tables, and runtime screens

Introduction

Converting a PL7 application into a Unity Pro application generates certain correspondences which are described in the following paragraphs.

Printouts

No conversion of the printed parts (title pages, headers, and footers) is possible between PL7 and Unity Pro, because, in PL7, no page setup file is saved on the disk after the generation of a document.

Animation tables

The animation tables are automatically converted into Unity Pro format by the PL7 application converter.

The list of PL7 variables contained in each animation table is replaced by the corresponding list of Unity Pro variables.

NOTE: if a PL7 variable has no Unity Pro equivalent, an error will be signaled in the conversion report file

Runtime screens

The animation tables are automatically converted into Unity Pro format by the PL7 application converter.

The list of PL7 variables contained in each runtime screen is replaced by the corresponding list of Unity Pro variables.

NOTE: if a PL7 variable has no Unity Pro equivalent, an error will be signaled in the conversion report file

NOTE: the files which describe the runtime screens are contained in the sub-folders whose addresses are specified in the .fef source file. The conversion procedure from PL7 to Unity Pro maintains this structure.

Differences between PL7 and Unity Pro

IV

Subject of this Part

This part presents the main differences between PL7 programming and its equivalent in Unity Pro.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
13	Differences between the application structures	165
14	Differences between common language elements	167
15	Differences between Structured Text language elements	183
16	Differences between Instruction List language elements	185
17	Different display in runtime screens	187

Differences between the application structures

13

13.1 Differences between PL7 and Unity Pro: functional modules

Functional Modules

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Functional modules

The PL7 application converter converts only the functional models which:

- contain a complete Grafcet section (PRL, Chart, Pos),
- do not contain isolated elements of Grafcet,
- do not contain macro steps.

NOTE: a functional model containing a Grafcet function, even it is complete, is not converted if it also contains a macro step.

The names of the functional modules which cannot be converted are listed in the conversion report file.

Differences between common language elements

14

Subject of this Chapter

This chapter describes the main differences between the objects common to different languages.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
14.1	Differences between PL7 and Unity Pro: types and tables	168
14.2	Differences between PL7 and Unity Pro: objects	171
14.3	Differences between PL7 and Unity Pro: instructions and functions	176
14.4	Differences between PL7 and Unity Pro: SFBs	181
14.5	Differences between PL7 and Unity Pro: Timers	182

14.1 Differences between PL7 and Unity Pro: types and tables

Subject of this Section

This section describes the main differences between PL7 types and tables and their Unity Pro equivalent.

What's in this Section?

This section contains the following topics:

Topic	Page
Types and tables	169
Operations between mixed types	170

Types and tables

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Types

WORD and DWORD types exist in Unity Pro but do not make it possible to carry out arithmetic operations.

Arithmetic and logic operations are performed in Unity Pro with INT and DINT type objects.

For this reason the PL7 converter converts WORD or DWORD type objects into INT or DINT type objects.

Example: The SHL operation on WORD (or DWORD) type objects is converted into SHL_INT (or SHL_DINT) *(see page 95)*.

Tables

The PL7 converter modifies the syntax of the declaration in the tables (Boolean, word, double word, floating point and byte tables).

Example: `ARRAY1:%M1:10` is converted into`ARRAY1:ARRAY[0..9] OF BOOL`.

NOTE: in Unity Pro, elements keep the same name as in PL7 (**example:** ARRAY1).

Specific types and tables

(WORD) words and (DWORD) double words in TIME, DATE, TOD and DT formats are converted into INT and DINT.

Example: `%MD10:=ADD_TOD(%MD20,%MD30)` is converted into:

```
%MD10:=ADD_TOD_PL7(%MD20,%MD30).
```

Time management EFs in PL7 are converted into their corresponding EFs in Unity Pro *(see page 95)*.

Operations between mixed types

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Words and double words

Operations between different object types are not possible in Unity Pro.

During the conversion procedure, operations of this type are detected and replaced by their corresponding EFs *(see page 95)*.

Example: `MD0:=%MD0+%MW4` is converted into

```
%MD0:=%MD0+INT_TO_DINT(%MW4).
```

Example: `%MW20:=%MD10` is converted into

```
%MW20:=DINT_TO_INT(%MD10).
```

Tables and (double) words

Operations between word tables or double word tables and words or double words are not possible in Unity Pro.

During the conversion procedure, operations of this type are detected and replaced by their corresponding EFs *(see page 109)*.

Example: `%MW100:20:=%MW10+5` is converted into

```
MOVE_INT_ARINT(%MW10+5,%MW100:20).
```

Example: `%MW100:20:=%MW100:20+5` is converted into

```
%MW100:20:=ADD_ARINT_INT(%MW100:20.5).
```

Assignment of tables

The assignment of a word or double word table into a bit table is not possible in Unity Pro.

During the conversion procedure, the assignment operator `:=` is replaced by the corresponding EF *(see page 100)*.

Example: `%M0:16:=%MW20` is converted into

```
MOVE_INT_ARX(%MW20,%M0:16).
```

14.2 Differences between PL7 and Unity Pro: objects

Subject of this Section

This section describes the main differences between PL7 objects and their Unity Pro equivalent.

What's in this Section?

This section contains the following topics:

Topic	Page
Immediate values	172
Memory objects (variables and constants)	173
Word bits	174
Symbolized tables and indexed objects	175

Immediate values

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Integer and long integer words

WORD and DWORD types are converted into INT and DINT types by the PL7 converter.

A hexadecimal immediate value greater than 16#7FFF (+32767) is converted to a negative decimal value, the original value is shown in the comment (example: `%MW0:=16#ABCD` is converted to `%MW0:=-21555 {16#ABCD}`).

The conversion of the immediate value of a long integer is identical for values greater than 16#7FFFFFFF (+2147483647) (example: `%MD80:=16#ABCDABCD` is converted to `%MD80:=-1412584499 {16#ABCDABCD}`).

Network address words

The address `ADR#` is replaced in Unity Pro by an EF. For a given address `ADR#{r.s}\xy.i.c\xy.i.SYS`, there are two possible scenarios:

- xy corresponds to the bus address:
  - the converter replaces xy by a bus number,
  - the converter chooses the bus number, which is specified in the Unity Pro configuration.
- xy corresponds to the rack number and position:
  - the converter replaces xy by r.m,
  - the rack 0 information becomes explicit.
    If xy < 100, the rack number is equal to 0 (example: xy=12, r.m = 0.12).
    If xy > 100, the rack number is equal to the first digit (example: xy=715, r.m = 7.15).

Memory objects (variables and constants)

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Character strings

The converter replaces the %MB and %KB objects with character strings; their position in the memory remains identical.

The name associated with the new character string is:

- if no symbol is associated with the object %MB or %KB, **MBi_l**:
  - i = integer,
  - l = length of character table,

  (example: %MB1000:20 is converted to MB1000_20:STRING[20]).
- if a symbol is associated with the object %MB or %KB, **symbole_l**:
  - symbol = symbol of variable,
  - l = length of character table,

  (example: %MB1000:20 with the symbol TABLE is converted into TABLE_20:STRING[20]).

If there is a comment associated with the first element of a table from %MB or from %KB, the converter declares a character string of length 1 and associates the comment with this (example: TABLE, symbol of %MB500:20 (*This is the TABLE comment*), is converted to TABLE_1:STRING[1](*This is the TABLE comment*)).

Word bits

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Extracted bit

In order to avoid any conflicts of syntax in the PL7 user applications to be converted, the Unity Pro syntax for extracted bits has been modified.

The PL7 syntax for the extracted bit `:Xi` is therefore replaced with Unity Pro syntax `.i`.

Example: `%IW12.3.1:X5` is converted into `%IW12.3.1.5`.

The Unity Pro syntax for the extracted bit on the rank 0 object is as follows: **%IW12.3.0.5**.

Symbolized tables and indexed objects

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Symbolized tables

In Unity Pro you can associate a symbol with a simple object, but cannot use the same symbol to refer to a table.

Example: if `%MWi` is symbolized by `TABA`, in PL7 `%MWi:L` is symbolized by `TABA:L`.

This is no longer possible in Unity Pro.

The PL7 application converter replaces `TABA:L` by a long integer table `L` named `TABA_L` and located on the basis of `%MWi`.

Symbolized indexed objects

In Unity Pro you can associate a symbol with a simple object, but cannot use the same symbol to refer to an indexed object.

Example: if `%MWi` is symbolized by `TABA`, in PL7 `%MWi[j]` is symbolized by `TABA[j]`.

This is no longer possible in Unity Pro.

The PL7 application converter replaces `TABA[j]` by a maximum length integer table named `TABA_AR` and located on the basis of `%MWi`.

NOTE: symbolized indexed input/output objects cannot be converted into equivalent tables as the length of this table is unknown. These objects are therefore converted into their non-symbolized form (address).

14.3 Differences between PL7 and Unity Pro: instructions and functions

Subject of this Section

This section describes the main differences between PL7 instructions and functions, and their Unity Pro equivalent.

What's in this Section?

This section contains the following topics:

Topic	Page
Table instructions and functions	177
Process control, Other and Communication instructions	179

Table instructions and functions

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Integer and long integer word tables: instructions

The operators (+, -, *, /, REM) between two integer and long integer word tables have been deleted in Unity Pro.

The PL7 application converter replaces these operators with EF equivalents *(see page 109)*.

Example: `TABINT1`, `TABINT2`, `TABINT3` are integer tables.
`TABINT1:=TABINT2 + TABINT3` is replaced with
`TABINT1:=ADD_ARINT(TABINT2,TABINT3).`

The operators (+, -, *, /, REM) between a table and an integer or long integer word have been deleted in Unity Pro.

The PL7 application converter replaces these operators with EF equivalents *(see page 109)*. A single EF is used for each switching operator.

Example: `INT1` is an integer; `TABINT1` and `TABINT2` are integer tables.

Both `TABINT1:=INT1 + TABINT2` and `TABINT1:=TABINT2 + INT1` are replaced with `TABINT1:=ADD_ARINT_INT(TABINT2,INT1)`

Integer and long integer word tables: logic instructions

The operators (AND, OR, XOR, NOT) between two integer and long integer word tables have been deleted in Unity Pro.

The PL7 application converter replaces these operators with EF equivalents *(see page 112)*.

Example: `TABINT1`, `TABINT2`, `TABINT3` are integer tables.
`TABINT1:=TABINT2 AND TABINT3` is replaced with
`TABINT1:=AND_ARINT(TABINT2,TABINT3).`

The operators (AND, OR, XOR) between a table and an integer or long integer word have been deleted in Unity Pro.

The PL7 application converter replaces these operators with EF equivalents *(see page 112)*. A single EF is used for each switching operator.

Example: `INT1` is an integer; `TABINT1` and `TABINT2` are integer tables.

Both `TABINT1:=INT1 AND TABINT2` and `TABINT1:=TABINT2 AND INT1` are replaced with `TABINT1:=AND_ARINT_INT(TABINT2,INT1)`

Tables: functions

The table functions for which you must indicate a rank (rank of an element in the table) behave in the same way as the corresponding PL7 functions, except when the rank is negative. In this case operation is as follows:

Functions	Operation when the rank is negative
COPY_ARDINT_AREBOOL	If one of the ranks is negative (source or destination), the function is not executed, and the resulting table is not modified. In PL7, the ranks are set automatically to 0 and the function is executed.
COPY_ARINT_AREBOOL	
COPY_AREBOOL_ARDINT	
COPY_AREBOOL_ARINT	
COPY_AREBOOL_AREBOOL	
EQUAL_***	If the rank from which the comparison is launched is negative, the result is equal to this negative rank and the function is not executed. In PL7, the function is executed from rank 0.
FIND_EQP_***	If the rank from which the search is launched is negative, the result is equal to this negative rank and the function is not executed. In PL7, the function is executed from rank 0.

Process control, Other and Communication instructions

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

IN, OUT, INOUT parameters

In Unity Pro, for all EFs that use input, output and input/output parameters (IN, OUT and INOUT), the order of these parameters has been modified.

Example: in PL7, the order of the parameters is as follows: **IN, OUT, INOUT**.

In Unity Pro, this becomes: **IN, INOUT, OUT**.

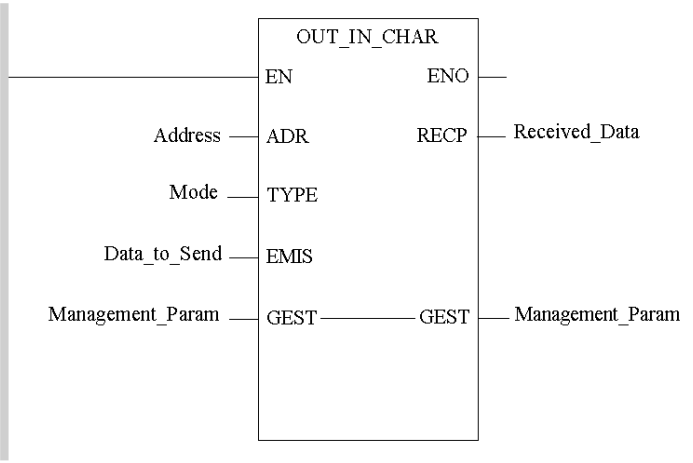
The EFs concerned by this modification are the following:

Process control instructions *(see page 119)*		**Other instructions** *(see page 120)*		**Communication instructions** *(see page 121)*	
PL7	**Unity Pro**	**PL7**	**Unity Pro**	**PL7**	**Unity Pro**
PID	PID_INT	FTOF	FTOF	DATA_EXCH	DATA_EXCH
PWM	PWM_INT	FTON	FTON	INPUT_CHAR	INPUT_CHAR
SERVO	SERVO_INT	FTP	FTP	OUT_IN_CHAR	OUT_IN_CHAR
		SCOUNT	SCOUNT	PRINT_CHAR	PRINT_CHAR
		FPULSOR	FPULSOR	RCV_TLG	RCV_TLG
				READ_ASYN	READ_ASYN
				READ_GDATA	READ_GDATA
				READ_VAR	READ_VAR
				SEND_REQ	SEND_REQ
				SERVER	UNITE_SERVER

NOTE: When you used a TSX series 7 PLC's PL7-2 or PL7-3 with the function DATA_EXCH, in reception, APP.NUM is interpreted as APP and the message with this address is managed by the first DATA_EXCH executed after reception.

NOTE: FTOF, FTON and FTP are not converted for M340 targets.

The following diagram shows the example of the communication instruction OUT_IN_CHAR.



The following table describes the parameters of the communication instruction OUT_IN_CHAR.

Input parameters (IN)	Input/output parameters (INOUT)	Output parameters (OUT)
ADR	GEST	RECP
TYPE		
EMIS		

The ST representation of the instruction OUT_IN_CHAR in PL7 is as follows:

```
OUT_IN_CHAR (Address, Mode, Data_to_Send, Received_Data,
Management_Param).
```

The ST representation of the instruction OUT_IN_CHAR in Unity Pro is as follows:

```
OUT_IN_CHAR (Address, Mode, Data_to_Send, Received_Data,
Management_Param).
```

14.4 Differences between PL7 and Unity Pro: SFBs

Types of Unity Pro EFB instances

Introduction

The conversion of a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Value of current step

The `PL7_DRUM` EFB has an extra output parameter compared with its PL7 equivalent. This parameter contains the value of the current step.

Henceforth, the values of the current step are assigned in the user program to DRUM output objects.

Example in structured text: `RESET %DRi` is replaced by:

```
RESET_PL7_DRUM (DRi)
WORD_TO_BIT (INT_TO_WORD (DRi.W),
             %M4,
             %O2.3,
             %O4.5,
             %M6, , , , , , , , , , , , )
```

NOTE: in the case of converting the ladder language, if the network including the call to DRUM contains the output objects of this DRUM, an error message is displayed. If these objects are assigned in the network, you must modify the program manually using the Converror message *(see page 49)* in the output window.

14.5 Differences between PL7 and Unity Pro: Timers

Differences between PL7 and Unity Pro: Timers

Introduction

This part explains the differences between the PL7 timers and IEC timers available in Unity Pro. Those differences are useful especially when migrating from PL7 software to Unity Pro.

The PL7 timers include:

- PL7_TON
- PL7_TOF
- PL7_TP

The IEC timers include:

- TON
- TOF
- TP

IEC timers and PL7 timers comparison

The table below lists the differences between IEC timers and PL7 timers:

	IEC Timers	**PL7 Timers**
Allowed amount of timers in application	Unlimited	255
Programming	• No time base • Preset time is a TIME type (t#1mn20s) • The current value of the timer is directly one output ET of TIME type	• Time base (min, sec, or ms) must be defined with an integer • The preset time is an INT type (time base factor) • The current value is provided by a public variable
Running Mode	The TON instance must be scanned in the program to refresh the output	The output is set once the timer is launched
Online Mode	Online modifications (add/modify) allowed on TON instance	The user cannot: • Modify the time base • Create new instance

Differences between Structured Text language elements

15

15.1 Differences between PL7 and Unity Pro: Structured Text language instructions

Command instructions

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

FOR instruction

The PL7 application converter converts the FOR command instruction.

In Unity Pro the index variable used in the FOR instruction is usable only within the programming loop.

During conversion of the first FOR instruction of a PL7 application , an error message will warn you to check that each of the programming loop variables is used only within the loop.

If this is not the case, the variable not having been declared, the command is not carried out.

Differences between Instruction List language elements

16

16.1 Differences between PL7 and Unity Pro: Instruction List language instructions

Boolean instructions

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

MPS, MRD, MPP instructions

The instructions MPS, MRD and MPP do not exist in Unity Pro. The PL7 application converter replaces these instructions with variables adapted to the environment.

For each module of a PL7 application in which an MPS, MRD or MPP instruction appears, the converter declares an 8 BOOL table and an INT type index simulating the behavior of the instruction to be replaced.

NOTE: before replacing MPS, MRD and MPP instructions, the PL7 applications converter verifies that the names of the created variables have not already been used in the application.

Different display in runtime screens

17

Runtime screens

Introduction

Converting a PL7 application into a Unity Pro application generates certain differences, which are described in the following paragraphs.

Runtime screens

The display format of the values of text objects (Binary and Hex) have been modified in the Runtime Screen tool.

The new format is identical to the one used in all other tools of Unity Pro: animation tables and language editors.

Example: The WORD type variable, 0 in binary format `2#0000000000000000` is converted into `2#0000_0000_0000_0000`.

The PL7 applications to be converted may therefore show display errors. If the display zone has been adjusted to the maximum size of the text, in animation, running Unity Pro the value of the variable will be replaced by the sequence `###############`, indicating that the size of the zone is insufficient to display the value in its entirety.

You must then either resize the display zone of the text, or lower the size of the font used.

Appendices

Recommendations

A

Recommendations during conversion

At a Glance

When converting PL7 applications, certain cases have been identified in which manual solutions are required. These are described in this appendix.

⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR - INCOMPLETE CONVERSION

If you are using control loops parametered using PL7 variables (%MW, %MF...) in the FEF file to be converted, you must enter them again using Unity Pro.

Failure to follow these instructions can result in injury or equipment damage.

What to do when the conversion fails

The following table shows the procedure when the conversion fails just after opening the FEF file.

If	Then
an error message appears	consult the report file to obtain additional information.
in the report file, the error concerns an unknown hardware configuration	Then • open the STX file using version ≥4.3 of PL7, • modify the processor version, • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
in the report file, the error concerns the name of a character string longer than 32 characters	Then • using PL7, modify the name of the variable to lower the size to under 32 characters, • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.

If	Then
in the report file, the error concerns a name conflict between variables, sections, etc.	Then ● using PL7, modify the names or symbols concerned, ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.
the FEF file was created with a version of PL7 previous to 4.0	Then ● open the FEF file using version ≥4.3 of PL7 *(see page 18)*, ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.
the FEF file was created with a 4.0, 4.1 or 4.2 version of PL7	Then ● open the FEF file using version ≥4.3 of PL7 *(see page 18)*, ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.
the processor is not level 3	Then ● open the FEF file using version ≥4.3 of PL7, *(see page 18)* ● modify the processor version *(see page 18)*, ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.
in the FEF file, built-in control loops are used, set using PL7 variables (%MF, for example)	Then ● using PL7, modify the loops concerned, ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.
you have modified the configuration of a TSX SCY 21601 module just before exporting the FEF file	Then ● using PL7, save the stx file before exporting the application (this allows you to save all last-minute modifications), ● export the PL7 application to obtain a new FEF file, ● restart the conversion, opening the FEF file with Unity Pro.

What to do when the import fails

The following table describes the procedure when the import phase fails (this phase is launched automatically after the conversion phase).

If	Then
the hardware configuration of the FEF file has ATV 16 controllers on a Fipio bus	Then • using PL7 remove the ATV 16 controllers from the Fipio bus (ATV 16 no longer recognized by Unity), • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
the hardware configuration of the FEF file has CCX 17 controllers on a Fipio bus	Then • using PL7 remove the CCX 17 controllers from the Fipio bus (CCX 17 no longer recognized by Unity), • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
the hardware configuration of the FEF file has SIMULATION devices on a Fipio bus	Then • using PL7 remove the SIMULATION devices from the Fipio bus (SIMULATION no longer recognized by Unity), • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
the hardware configuration of the FEF file has a TSX P57 2823 or TSX P57 4823 processor	Then • using PL7 replace the processor with one that is recognized by Unity, then modify the configuration to obtain equivalent functionalities (for example, use a TSX P57 253 processor and ETY module to replace a TSX P57 2823), • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
the PL7 application to convert has a TSX SPY 400 simulation module or TSX ISPY 100 weighing module	Then • using PL7, delete all the parts concerning these (configuration, sections, etc.), • export the PL7 application to obtain a new FEF file, • restart the conversion, opening the FEF file with Unity Pro.
an indexed variable is used as an activation condition	using Unity Pro, write the missing condition without using indexed variables.
A simulation module is used in the PL7 Pro configuration during import	Always remove the simulation module from the PL7 Pro configuration from the rack and when used with FIP I/O bus.

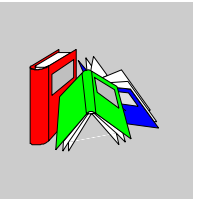
What to do when the generation of the project fails

The following table describes the procedure when the Unity Pro generation phase fails.

If, in the generation report, one of the following cases appears	Then
a section written in ST language has a truncated line	Then ● using Unity Pro, correct the faulty line return generated during editing (the ST editor does not authorize lines of over 300 characters), ● generate the project again.
an error occurs on a macro step bit	Then ● using Unity Pro, check that the macro step is declared (using PL7 it is possible to use a macro step bit without it being declared), ● if this is the case, correct the program accordingly. ● generate the project again.
an error on input or output-type objects (%Q1.2.48:16 for example) is generated	Then ● check that these objects are configured using PL7, ● if this is not the case, correct the program accordingly. ● generate the project again.
an error on FIPIO objects (%IW\2.80\0.0.0.3 for example) is generated	Then ● check that these objects are configured using PL7, ● if this is not the case, correct the program accordingly. ● generate the project again, ● if you are using ADM 390 10 modules, it is no longer possible to use FIPIO objects under Unity Pro.
an error in the system word tables is generated	Then ● create a table of integers with 4 elements, ● assign each element one by one. ● generate the project again.
an error occurs on a table-type DFB output	a table-type DFB output is no longer accessible outside the DFB. To access it, you must use the variable connected to the corresponding output pin. For further information see on DFBs, conversion precautions *(see page 38)*

If, in the generation report, one of the following cases appears	Then
an error on the DINT-type variables recognized as INT by Unity Pro	Then ● modify the program to make the types compatible, ● example: SD52, the solution consists of creating a DINT-type SD52 object by writing SD52:=INT_AS_DINT(%SW52,%SW53);. ● generate the project again.
an error occurs on the %I or %IW assigned in the program	modify the program to delete these assignments. Assignment on inputs is no longer accepted by Unity Pro.
an error occurs on a vertical comparison block located in the first column of the editor	using the **Copy/Paste** function, modify the program to move this block by one column.
an error occurs in an SFC. An error message indicates that alternative divergences or parallel convergences must be followed respectively by a transition or a step	modify the SFC following the indications of the output window.
an error indicates that the maximum number of configured steps is insufficient	increase the maximum number of steps as indicated in the output window (use the command **Tools** → **Project settings** →**Language extensions**). The calculation of the maximum number of steps using Unity Pro includes macro-steps.
an error indicates an incompatibility between the different types of an assignment (probably a multi-assignment using PL7)	Modify the program to remove these incompatibilities.
an error occurs on a multi-assignment in an LD operate block	modify the program by writing as many blocks as there are assignments.

Glossary



D

DFB

User Function Block.

DT

Date and Time.

E

EF

Elementary Function.

EFB

Elementary Function Block.

EVT

Event.

I

IL

Instruction List language.

L

LD

Ladder language.

S

SFB

Standard Function Block.

SFC

Sequential Function Chart language.

SR

Subroutine.

ST

Structured Text language.

STRING

Character string.

Index

A

ADR, *71*
analyzing an application, *45*
animation tables, *161*

C

character strings, *173*
converror, *46*
conversion settings for Grafcet elements, *159*
conversion wizard for PL7 Micro), *33*
converter, *49*
converting a PL7 DFB, *37*
converting a PL7 diagnostic DFB, *38*
converting an application, *27*
correspondences, *51*

D

direct addresses
    conversion wizard, *35*
directly addressed variables
    symbolization, *81*
disabling the protection, *28*

E

exporting a PL7 application, *28*
exporting a PL7 DFB, *38*

F

FAQs, *191*
FOR instruction , *183*
functional modules
    correspondences, *59*

G

Grafcet, *157*

H

hardware equivalences, *53*

I

importing a PL7 DFB, *43*
Instruction List
    correspondences, *147*
    differences, *185*
instructions
    correspondences, *95*
instructions
    differences, *176*

L

Ladder, *137*
language objects
    correspondences, *70*
    differences, *171*
LD sections
    split into small sections, *63*

N

network address, *72*

O

object types
    correspondences, *66*
    differences, *168*

P

PL7 Micro)
    conversion wizard, *33*
Processor equivalences, *53*

R

runtime screens, *161*
    correspondences, *161*
    differences, *187*

S

sections
    LD section splitting, *63*
SFBs
    correspondences, *127*
    differences, *181*
    in Modicon M340 applications, *129*
structural elements, *59*
Structured Text
    correspondences, *143*
    differences, *183*
symbolized directly addressed variables, *81*

T

topological addresses
    conversion wizard, *35*